Milan Hladík

Celočíselné Programování

text k přednášce

23. května 2017

Tento text zatím slouží jako doprovodný text přednášky Celočíselné programování na studiu informatiky Matematicko-fyzikální fakulty Univerzity Karlovy v Praze.

Autor čerpal především z knih Nemhauser and Wolsey [1999]; Schrijver [1998]; Wolsey [1998]. Další klasické knihy jsou např. Korbut and Finkelstein [1971]; Taha [1975]. O historii pěkně pojednává soubor esejů Grötschel [2012].

Díky Tomáši Pokornému za některé obrázky, opravy chyb a doplnění několika důkazů.

Případné připomínky a chyby zasílejte prosím na adresu hladik@kam.mff.cuni.cz.

Obsah

Část I

Teorie a metody

Kapitola 1

Teorie

1.1 Úvod

V celočíselném programování se střetávají dva odlišné světy: spojitá a kombinatorická optimalizace. Střetávají se dvojím způsobem. Jednak se formulaci problému, kde se vyskytují jak spojité proměnné a funkce, a jednak v metodách na řešení, kdy moderní algoritmy kombinují techniky z obou disciplín.

Celočíselné programování zažilo v minulých dekádách intensivní rozvoj a je stále oblastí s intensivním výzkumem. Je to způsobeno tím, že obrovská řada praktických problémů se dá zformulovat jako celočíselný program. Přitom je celočíselné programování těžké na řešení, takže představuje výzvu pro další výzkum a vývoj a má potenciál na další posun.

Pár slov ke klasifikaci úloh celočíselného programování:

Celočíselné programování

Úlohou celočíselného programování se rozumí úloha

$$\begin{aligned} \max\ f(x)\ \text{ za podm. }\ & g_j(x) \le 0 \quad \forall j = 1, \dots, m, \\ & x_i \in \mathbb{Z} \quad \forall i \in C, \end{aligned}$$

kde $C \subseteq \{1, \dots, n\}$. Jedná se tedy o úlohu matematického programování s podmínkami na celočíselnost některých proměnných. Pokud $C = \{1, \dots, n\}$, jde o čistou úlohu celočíselného programování (všechny proměnné celočíselné), jinak ji nazýváme smíšenou.

Celočíselnost proměnných jde přeformulovat jako nelineární podmínka. Například pro binární proměnné takto:

$$x \in \{0, 1\} \quad \iff \quad x \in [0, 1] \ \wedge\ x(1 - x) = 0.$$

Nicméně, celočíselné proměnná mají svá specifika, proto je výhodnější se jimi zabývat zvlášť.

Lineární celočíselné programování

Jsou-li funkce $f(x), g_j(x)$ lineární, pak jde o lineární celočíselné programování

$$\begin{aligned} \max\ c^T x\ \text{ za podm. }\ & Ax \le b \\ & x_i \in \mathbb{Z} \quad \forall i \in C. \end{aligned}$$

V tomto textu se budeme věnovat výhradně tomuto lineárnímu případu. Uvidíme, že i tato formulace je dost obecná a pokrývá širokou třídu úloh. Navíc techniky, které se naučíme pro lineární úlohy, se používají i pro nelineární úlohy.

0-1 programování

Speciálním typem lineárního celočíselného programu je 0-1 program s binárními proměnnými

$$\max\ c^T x\ \text{ za podm. }\ Ax \le b, \quad x \in \{0, 1\}^n.$$

Na druhou stranu, zas tak speciálním typem není. Předpokládejme omezenost množiny přípustných řešení (srov. poznámka 1.24), tedy existuje $u > 0$ tak, že každé přípustné řešení x splňuje $|x_i| \leq u$, $\forall i$. Pak každou proměnnou x_i nahradíme binárním rozvojem $x_i = \sum_{k=0}^{\lfloor \log(u) \rfloor} 2^k y_{ki}$, kde $y_{ki} \in \{0, 1\}$ jsou binární proměnné. Celkový počet proměnných vzrostl $\lfloor \log(u) \rfloor$-krát.

Značení

Pro reálné číslo r používáme $\lfloor r \rfloor = \max\{z \in \mathbb{Z};\ z \leq r\}$ pro jeho (dolní) celou část, $\lceil r \rceil = \min\{z \in \mathbb{Z};\ z \geq r\}$ pro horní celou část a $\{r\} = r - \lfloor r \rfloor$ pro zlomkovou část.

1.2 Celočíselný polyedr

Mějme $A \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{Q}^m$. Tento předpoklad sice je na újmu obecnosti (srov. poznámka 1.2), ale z praktického hlediska se vesměs setkáváme jen s racionálními daty, a reprezentovat na počítači můžeme také jen (některá) racionální čísla.

V této kapitole se nadále budeme zabývat čistou úlohou lineárního celočíselného programování

$$\max\ c^T x \ \text{ za podm. } \ x \in M \cap \mathbb{Z}^n, \tag{1.1}$$

kde

$$M = \{x \in \mathbb{R}^n;\ Ax \leq b,\ x \geq 0\}.$$

Bez podmínek celočíselnosti dostaneme lineární program

$$\max\ c^T x \ \text{ za podm. } \ x \in M. \tag{1.2}$$

Jeho optimální hodnota dává horní odhad na optimální hodnotu (1.1), ale z optimálního řešení určit optimum původní úlohy není jednoduché. Úloha (1.2) se nazývá *lineární*, nebo též *celočíselná relaxace* (celočíselná relaxace protože došlo k „uvolnění“ celočíselných podmínek a lineární relaxace, protože jsme dostali lineární podmínky).

Označme

$$M_I \coloneqq \operatorname{conv}\{x \in \mathbb{Z}^n;\ Ax \leq b,\ x \geq 0\}$$

konvexní obal množiny přípustných řešení. Zřejmě platí $(M \cap \mathbb{Z}^n) \subseteq M_I \subseteq M$. Úloha (1.1) se pak redukuje na lineární program

$$\max\ c^T x \ \text{ za podm. } \ x \in M_I.$$

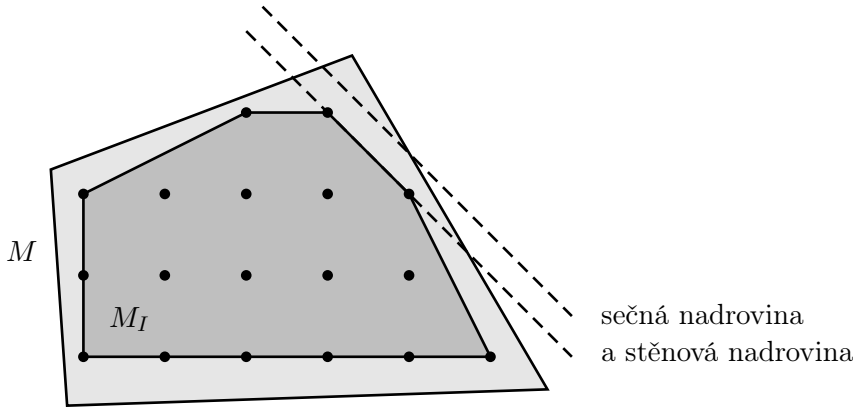
Potíž je, že sestrojit množinu M_I je velmi těžké, až na speciální případy probírané v sekci 1.3. Můžeme se každopádně aspoň k M_I přiblížit tím, že množinu M budeme ořezávat přidáváním lineárních podmínek, které odstraní kus M, ale neodříznou nic z M_I. Takovéto podmínky se nazývají *řezy* nebo také *sečné nadroviny* v lineárním případě a tvoří základ metod probíraných v kapitole 2. Sečná nadrovina, která určuje facetu (stěnu dimenze $n-1$) polyedru M_I se nazývá *stěnová* (ang. *facet-defining*), viz obrázek 1.1.

Následující věta potvrzuje, že M_I je konvexní polyedr, bez ohledu na to, je-li v omezení rovnice nebo nerovnice. Pro M omezené je to zřejmé z duálně nerovnicovo-vrcholového popisu polyedru, ale pro neomezený případ to zřejmé na první pohled není.

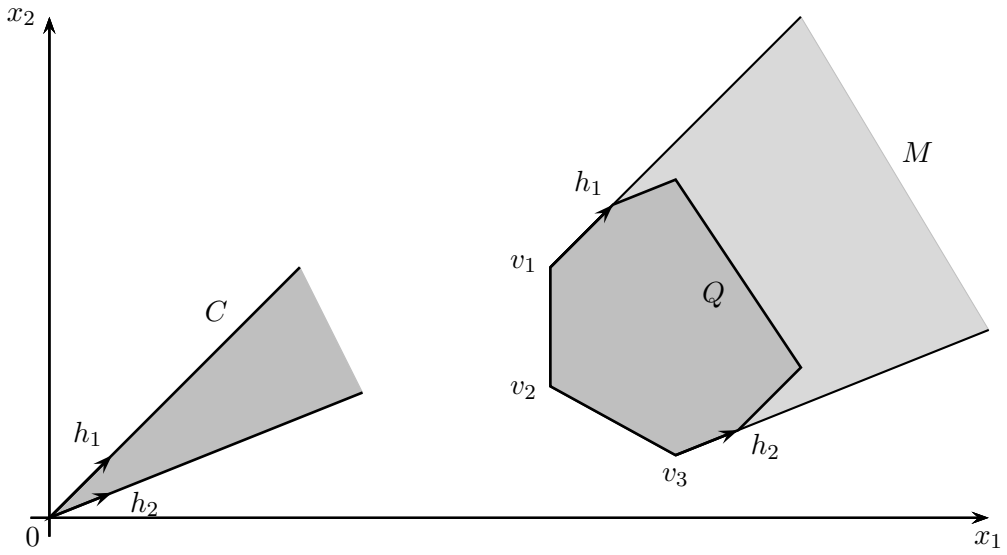
Věta 1.1 (Meyer, 1974)**.** *M_I je konvexní polyedr.*

Důkaz. To, že M_I je konvexní, plyne z jeho definice. Zbývá tedy dokázat, že je to polyedr. Buďte v_i, $i \in I$, vrcholy a h_j, $j \in J$, směry neomezených hran M. Protože v_i a h_j mají racionální složky, můžeme bez újmy na obecnost předpokládat, že $h_j \in \mathbb{Z}^n$ (čehož docílíme přenásobením nejmenším společným násobkem jmenovatelů). Každý bod $x \in M$ lze psát ve tvaru

$$x = \textstyle\sum_{i \in I} \alpha_i v_i + \sum_{j \in J} \beta_j h_j = \sum_{i \in I} \alpha_i v_i + \sum_{j \in J} \lfloor \beta_j \rfloor h_j + \sum_{j \in J} \{\beta_j\} h_j \tag{1.3}$$



Obrázek 1.1: Konvexní polyedr M, celočíselné body v něm a jejich konvexní obal M_I. Sečná a stěnová sečná nadrovina.



Obrázek 1.2: Rozdělení $M = C + Q$, důkaz věty 1.1.

pro určité $\alpha_i \geq 0$, $\sum_{i\in I} \alpha_i = 1$, $\beta_j \geq 0$. Označme

$$Q := \left\{ \sum_{i\in I} \alpha_i v_i + \sum_{j\in J} \beta_j h_j;\ \alpha_i \geq 0,\ \sum_{i\in I} \alpha_i = 1,\ 0 \leq \beta_j \leq 1 \right\},$$
$$C := \left\{ \sum_{j\in J} \beta_j h_j;\ \beta_j \geq 0 \right\}.$$

Zde Q je omezený konvexní polyedr a C konvexní polyedrický kužel. Polyedr Q representuje omezenou část M a C jeho neomezené směry. Pak můžeme psát

$$M = Q + C := \{x + y;\ x \in Q,\ y \in C\}.$$

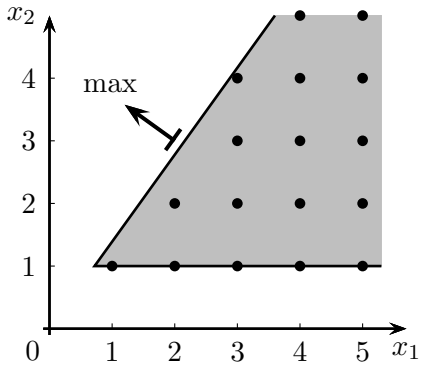
Vlastnost kužele C je ta, že $M + C = M$.

Naším cílem bude ukázat, že $M_I = Q_I + C$, viz obrázek 1.2.

„$\subseteq$“ Buď $x \in M \cap \mathbb{Z}^n$. Podle (1.3) je

$$x - \sum_{j\in J} \lfloor \beta_j \rfloor h_j = \sum_{i\in I} \alpha_i v_i + \sum_{j\in J} \{\beta_j\} h_j \in Q$$

celočíselný vektor, a tudíž $x \in (Q \cap \mathbb{Z}^n + C) \subseteq (Q_I + C)$. Tím pádem z konvexity $Q_I + C$ dostáváme $M_I = \mathrm{conv}(M \cap \mathbb{Z}^n) \subseteq (Q_I + C)$.



Obrázek 1.3: Nekonečný polyedr, poznámka 1.2.

„$\supseteq$“ Platí $Q_I + C \subseteq M_I + C = M_I + C_I \subseteq (M + C)_I = M_I$, srov. cvičení 1.2. □

Poznámka 1.2. Bez podmínek racionality A, b věta neplatí, např. pro

$$\max\ \alpha x_1 - x_2 \ \text{ za podm. } \ \alpha x_1 - x_2 \le 0,\ x_2 \ge 1,\ x \in \mathbb{Z}^2,$$

kde $\alpha > 0$ je iracionální, viz obrázek 1.3. V tom to případě M_I není konvexní polyedr, protože má nekonečně mnoho hran. Tato úloha navíc nemá optimální řešení, přestože je přípustná a omezená (tj., účelová funkce je omezená). Důvod je tem, že na stěně $\alpha x_1 - x_2 = 0$ se nenachází žádný celočíselný bod, ale celočíselnými body se ke stěně můžeme přiblížit libovolně přesně. Supremum účelové funkce je tudíž 0, ale hodnota se nenabyde. Takováto situace pro racionální data nastat nemůže.

Poznámka 1.3. Důsledek z důkazu věty: Je-li $M_I \neq \emptyset$, pak M a M_I mají stejný kužel neomezených směrů, a tudíž úloha (1.1) je neomezená právě tehdy, když je neomezená (1.2).

Cvičení

1.1. Zobecněte větu 1.1 na smíšenou úlohu, obsahující celočíselné i spojité proměnné. To znamená, dokažte, že konvexní obal množiny $\{x \in \mathbb{R}^m,\ y \in \mathbb{Z}^n;\ Ax + By \le c\}$ je konvexním polyedrem.

1.2. Rozhodněte zda pro konvexní polyedry P, Q platí $(P+Q)_I = P_I + Q_I$, popř. aspoň nějaká inkluze.

1.3 Unimodularita – kdy složité se stává jednodušším

Definice 1.4. Matice $A \in \mathbb{Z}^{m\times n}$ je *totálně unimodulární*, pokud každý její subdeterminant je -1, 0 či 1.

Nechť $A \in \mathbb{Z}^{m\times n}$ má hodnost m. Pak jí nazveme *unimodulární*, pokud každá její sloupcová báze (tj., regulární podmatice řádu n) má determinant ± 1.

Poznámka 1.5. Připomeňme některé věty pro práci s determinanty:

- *Laplaceův rozvoj podle k-tého sloupce:* $\det(A) = \sum_i (-1)^{i+k} a_{ik} \det(A^{ik})$, kde A^{ik} je matice A bez řádku i a sloupce k.
- *Cramerovo pravidlo:* Mějme soustavu rovnic $Ax = b$, kde A je regulární. Pak i-tá složka řešení je $x_i = \frac{\det(A_i)}{\det(A)}$, kde A_i je matice, která vznikne z A nahražením i-tého sloupce za vektor b.

Vztah mezi unimodularitou a totální unimodularitou vystihuje následující tvrzení.

Věta 1.6. *Buď $A \in \mathbb{Z}^{m\times n}$. pak A je totálně unimodulární právě tehdy, když $(A \,|\, I_m)$ je unimodulární.*

Důkaz. „$\Rightarrow$“ Buď B sloupcová báze $(A \,|\, I)$, kterážto matice je zřejmě celočíselná. Pak z Laplaceova rozvoje podle sloupců I_B dostaneme $\det(A_B \,|\, I_B) = \pm \det(A'_B) = \pm 1$, kde A'_B je matice vzniklá z A_B odstraněním řádků, kde I_B má jedničky.

„⇐“ Buď A' čtvercová podmatice A. Uvažme bázi B takovou, že A' je částí A_B a v řádcích, kde se nachází, I_B neobsahuje jedničky. Pak $\det(A') = \pm\det(A_B \mid I_B) = \pm 1$. Pokud by B nebyla báze, pak $(A_B \mid I_B)$ by byla singulární a $\det(A') = \pm\det(A_B \mid I_B) = 0$. Schematicky:

$$A' = \begin{pmatrix} a'_{1b_1} & \dots & a'_{1b_n} \\ \vdots & \ddots & \\ a'_{nb_1} & & a'_{nb_n} \end{pmatrix}, \quad (A_B|I_B) = \left(\begin{array}{ccc|cccccc} a_{1b_1} & \dots & a_{1b_n} & 1 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \ddots & & 0 & \ddots & 0 & 0 & 0 & 0 \\ a_{ib_1} & \dots & a_{ib_n} & 0 & 0 & 1 & 0 & 0 & 0 \\ a'_{1b_1} & \dots & a'_{1b_n} & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \ddots & & 0 & 0 & 0 & 0 & 0 & 0 \\ a'_{nb_1} & & a'_{nb_n} & 0 & 0 & 0 & 0 & 0 & 0 \\ a_{jb_1} & \dots & a_{jb_n} & 0 & 0 & 0 & 1 & 0 & 0 \\ \vdots & \ddots & & 0 & 0 & 0 & 0 & \ddots & 0 \\ a_{mb_1} & \dots & a_{mb_n} & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right)$$

□

Z hlediska celočíselného programování je stěžejní následující věta, která říká, že úlohy s (totálně) unimodulární maticí můžeme řešit lineárním programováním a celočíselnost řešení je zaručena.

Věta 1.7 (Hoffman & Kruskal, 1956)**.**

(1) Buď $A \in \mathbb{Z}^{m\times n}$ hodnosti m. Pak $M = \{x \in \mathbb{R}^n;\ Ax = b,\ x \geq 0\}$ má celočíselné vrcholy pro každé $b \in \mathbb{Z}^m$ právě tehdy, když A je unimodulární.

(2) Buď $A \in \mathbb{Z}^{m\times n}$. Pak $M' = \{x \in \mathbb{R}^n;\ Ax \leq b,\ x \geq 0\}$ má celočíselné vrcholy pro každé $b \in \mathbb{Z}^m$ právě tehdy, když A je totálně unimodulární.

Důkaz. (1) „⇒“ Buď B přípustná báze. Chceme dokázat, že $\det(A_B) = \pm 1$, neboli $\det(A_B^{-1}) = \det(A_B)^{-1} = \pm 1$. To je ekvivalentní s tvrzením, že A_B^{-1} má celočíselné složky: Je-li $A_B^{-1} \in \mathbb{Z}^{m\times m}$, pak z rovnosti $\det(A_B)\det(A_B^{-1}) = 1$ a celočíselnosti obou determinantů plyne $\det(A_B^{-1}) = \pm 1$. Naopak, je-li $\det(A_B^{-1}) = \pm 1$, tak adjungovaná matice $\operatorname{adj}(A_B)$ je celočíselná a tedy $A_B^{-1} = \frac{1}{\det(A_B)}\operatorname{adj}(A_B)$ jakbysmet.

Celočíselnost A_B^{-1} ukážeme po sloupcích. Uvažme i-tý sloupec $A_B^{-1}e_i$ a definujme $x \coloneqq y + A_B^{-1}e_i \geq 0$, kde $y \geq 0$ je celočíselný a dostatečně velký vektor. Nyní definujme $b \coloneqq A_B x = A_B(y + A_B^{-1}e_i) = A_B y + e_i \in \mathbb{Z}^m$. Definujme vektor x' tak, že $x'_B = x$ a $x'_N = 0$. Nyní x' je vrchol polyedru M a podle předpokladu je celočíselný. Tudíž $A_B^{-1}e_i = x - y$ je také celočíselný.

„⇐“ Buď x vrchol M, příslušný bázi B. Označme $N \coloneqq \{1, \dots, m\} \setminus B$. Pak $x_N = 0$ a $x_B = A_B^{-1}b$, neboli $A_B x_B = b$. Protože $\det(A_B) = \pm 1$, podle Cramerova pravidla jsou složky x_B celočíselné.

(2) Přepíšeme M' do tvaru $M' = \{x \in \mathbb{R}^n;\ Ax + Ix' \leq b,\ x, x' \geq 0\}$ a použijeme tvrzení 1.6 s bodem (1). □

Celočíselné programování s (totálně) unimodulárními maticemi je tedy výpočetně jednoduché. Celočíselnost můžeme relaxovat a lineární program automaticky najde celočíselné optimum. Zde je potřeba pouze podotknout, že pokud není optimum jednoznačné, ale tvoří jej celá stěna, pak je celočíselným optimem libovolný vrchol této stěny.

Důsledek 1.8. *Je-li A totálně unimodulární a $b \in \mathbb{Z}^m$, pak úloha celočíselného programování*

$$\max\ c^T x \quad \textit{za podm.}\ \ Ax \leq b,\ x \geq 0,\ x \in \mathbb{Z}^m$$

je ekvivalentní s úlohou lineárního programování

$$\max\ c^T x \quad \textit{za podm.}\ \ Ax \leq b,\ x \geq 0,\ x \in \mathbb{R}^m.$$

Ukážeme si dvě speciální třídy úloh, kde matice v omezeních je automaticky totálně unimodulární.

Definice 1.9. Buď $G = (V, E)$ neorientovaný graf. Pak *matice incidence* grafu G je matice $A_G \in \{0,1\}^{|V|\times|E|}$ s prvky $(A_G)_{ve} = 1$ pro $v \in e$, a s nulami jinak.

Buď $G = (V, E)$ orientovaný graf. Pak *matice incidence* grafu G je matice $A_G \in \{0, \pm 1\}^{|V|\times|E|}$ s prvky $(A_G)_{ve} = -1$ pro $e = (v, u)$, $(A_G)_{ve} = 1$ pro $e = (u, v)$, a s nulami jinak.

Věta 1.10. *Buď A matice incidence neorientovaného grafu G. Pak A je totálně unimodulární právě tehdy, když G je bipartitní graf.*

Důkaz. „⇒“ Bez újmy na obecnost předpokládejme, že G neobsahuje izolovaný vrchol, protože nulový řádek neovlivní unimodularitu matice A ani bipartitnost grafu G. Podle věty 1.7 má polyedr $\{x \in \mathbb{R}^{|V|};\ A^T x = e,\ x \geq 0\}$ celočíselné vrcholy. Protože je neprázdný (obsahuje vektor $\frac{1}{2}e$) a omezený (leží v nezáporném oktantu a součtem rovnic dostaneme omezení $a^T x = |V|$, kde $a > 0$), obsahuje celočíselné řešení x^*. Pak indexy s $x_i^* = 1$ odpovídají jedné partitě a ostatní té druhé.

„⇐“ Matematickou indukcí podle velikosti podmatice. Podmatice řádu 1 mají zřejmě determinant 0 nebo 1. Nyní uvažujme podmatici A' řádu k. Jsou-li v každém sloupci dvě jedničky, pak součet řádků, odpovídajících první partitě, se rovná součtu řádků druhé partity, a tudíž $\det(A') = 0$. V opačném případě obsahuje matice A' sloupec s maximálně jednou jedničkou, na ten aplikujeme Laplaceův rozvoj a použijeme indukční předpoklad. □

Příklad 1.11. Příkladem úlohy, kde se vyskytuje matice incidence bipartitního grafu je např. *dopravní problém*

$$\min \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \quad \text{za podm.} \quad \sum_{j=1}^{n} x_{ij} = a_i\ \forall i,\quad \sum_{i=1}^{m} x_{ij} = b_j\ \forall j,\quad x \geq 0,$$

kde a_i, $i = 1, \ldots, m$ jsou kapacity výrobců, b_j, $j = 1, \ldots, n$ kapacity spotřebitelů, a c_{ij} cena přepravy jednotky zboží od i-tého výrobce k j-tému spotřebiteli. Zde se ještě navíc předpokládá, aby $\sum_i a_i = \sum_j b_j$. Nazývá se podmínkou ekonomické rovnováhy, a je přirozenou podmínkou, neboť bez ní by úloha neměla řešení.

Setřídíme-li proměnné $x_{11}, \ldots, x_{1n}, x_{21}, \ldots, x_{2n}, \ldots, x_{mn}, \ldots, x_{mn}$, potom matice soustavy rovnic z omezení

$$\begin{pmatrix}
1 & \ldots & 1 & & & & & & & \\
 & & & 1 & \ldots & 1 & & & & \\
 & & & & & & \ldots & & & \\
 & & & & & & & 1 & \ldots & 1 \\
1 & & & 1 & & & & 1 & & \\
 & \ddots & & & \ddots & & \ldots & & \ddots & \\
 & & 1 & & & 1 & & & & 1
\end{pmatrix}$$

je totálně unimodulární. □

Následující pozorování prý nahlédl už Poincaré roku 1900.

Věta 1.12. *Matice incidence orientovaného grafu je totálně unimodulární.*

Důkaz. Analogicky důkazu věty 1.10. □

Příklad 1.13. Příkladem úloh, kde se vyskytuje matice incidence orientovaného grafu jsou *toky v sítích.* Klasická úloha nalezení maximálního toku v síti (orientovaném grafu) $G = (V, E)$

$$\begin{aligned}
\max\ v \quad \text{za podm.} \quad \sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} &= 0,\ \forall i \in V \setminus \{s,t\} \\
\sum_{j:(s,j)\in E} x_{sj} - \sum_{j:(j,s)\in E} x_{js} &= v, \\
\sum_{j:(t,j)\in E} x_{tj} - \sum_{j:(j,t)\in E} x_{jt} &= -v, \\
0 \leq x &\leq u,
\end{aligned}$$

kde $u\colon E \mapsto \mathbb{R}^+$ jsou kapacity hran, $s \in V$ je počáteční vrchol a $t \in V$ koncový. Přidáme-li do grafu symbolickou hranu (t, s), pak úlohu maximálního toku můžeme formulovat jako hledání maximálního

toku skrze přidanou hranu za podmínek, že pro každý vrchol platí podmínka rovnováhy (přítok = odtok). Je-li A matice incidence grafu G, pak tato úloha má tvar

$$\max\ x_{ts}\ \text{ za podm. }\ Ax = 0,\ 0 \le x \le u.$$

Podobnou úlohou je nalezení nejlevnějšího toku (*minimum-cost flow*)

$$\min \sum_{(i,j)\in E} c_{ij}x_{ij}\ \text{ za podm. }\ \sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ij} = b_i\ \forall i \in V,\ \ l \le x \le u,$$

kde $l, u\colon E \mapsto \mathbb{R}^+$ je dolní a horní mez na tok v jednotlivých hranách, c_{ij} cena přepravy jednotky zboží po hraně $(i,j) \in E$, a $b \in \mathbb{R}^{|V|}$ je vyrovnávací člen nabídky/poptávky. □

Příklad 1.14. Jiným příkladem úloh, kde se vyskytuje matice incidence orientovaného grafu je hledání *nejkratší cesty v grafu*. Uvažujme orientovaný graf $G = (V, E)$. Nechť $c\colon E \mapsto \mathbb{R}^+$ jsou váhy na hranách a nechť $s \in V$ je počáteční vrchol a $t \in V$ koncový. Úlohu můžeme formulovat jako hledání nejlevnějšího toku velikosti 1

$$\begin{aligned}\min \sum_{(i,j)\in E} c_{ij}x_{ij}\ \text{ za podm. }\ &\sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ij} = 0,\ \forall i \in V \setminus \{s,t\},\\ &\sum_{j:(s,j)\in E} x_{sj} - \sum_{j:(j,s)\in E} x_{sj} = 1,\\ &\sum_{j:(t,j)\in E} x_{tj} - \sum_{j:(j,t)\in E} x_{tj} = -1,\\ &0 \le x \le 1.\end{aligned}$$

Poznamenejme, že pro neorientované grafu je situace trochu jiná v tom, že pokud povolíme záporné váhy na hranách nebo pokud hledáme nejdelší cestu, tak se problém stává NP-těžkým. Poznamenejme, že pro nejdelší cestu nestačí změnit „min“ na „max“ ve formulaci úlohy, protože pak program nenajde nejdelší cestu, ale nejdelší tah. □

Poznamenejme, že totální unimodularita matice je invariantní vůči transpozici, výměně řádků, mazání řádků, zdvojování řádků, násobení -1, přidávání jednotkových a nulových řádků, konkatenace do blokově diagonálních matic atp.

Na konec této sekce si položme otázku jak je těžké rozhodnout, zda daná matice $A \in \{0, \pm 1\}^{m\times n}$ je totálně unimodulární? Tuto otázku lze rozhodnout v polynomiálním čase metodou Seymourovy dekompozice z roku 1980 [Schrijver, 1998]. Ta je založená na faktu, že A je totálně unimodulární právě tehdy, když lze vytvořit z tzv. síťových matic (zobecnění matice incidence orientovaného grafu) a několika pevných matic pomocí výše zmíněných úprav a jim podobných.

1.4 Složitost podrobně

Věta 1.15 (Cook, 1971)**.** *Celočíselné programování je NP-těžké.*

Důkaz. Na celočíselné programování lze převést řada NP-těžkých problémů, např.:

- Vrcholové pokrytí grafu $G = (V, E)$ lze zformulovat jako

$$\min \sum_{i\in V} x_{ij}\ \text{ za podm. }\ x_i + x_j \ge 1\ \forall\{i,j\} \in E,\ x_i \in \{0,1\}\ \forall i \in V.$$

- SAT, splnitelnost booleovských formulí v konjunktivní normální formě. Zde logickým proměnným přísluší binární proměnné, a každou klauzuli přepíšeme jako lineární nerovnici. Např. klauzuli $V_1 \vee V_2 \vee \neg V_3$ přepíšeme na $v_1 + v_2 + (1 - v_3) \ge 1$. □

Naším dalším cílem bude ukázat, že celočíselné programování je NP-úplný problém, tedy náleží do třídy NP. To je kupodivu ta těžší část důkazu, a musíme si pro to nejprve připravit půdu. Nebude to však jednoúčelová práce, neboť dostaneme v důsledku i jiné pěkné výsledky.

Definice 1.16 (Velikost zápisu)**.** Buď $r \in \mathbb{Q}$, $v \in \mathbb{Q}^n$ a $A \in \mathbb{Q}^{m\times n}$, a nechť $r = \frac{p}{q}$ je v základním tvaru, tj. p, q nesoudělné a $p \in Z$, $q \in \mathbb{N}$. Pak velikost jejich zápisu definujeme:

$$\begin{aligned}\sigma(r) &\coloneqq 1 + \lceil \log_2(|p| + 1)\rceil + \lceil \log_2(q + 1)\rceil,\\ \sigma(v) &\coloneqq n + \sum_i \sigma(v_i),\\ \sigma(A) &\coloneqq mn + \sum_{i,j} \sigma(a_{ij})\end{aligned}$$

Velikost zápisu odpovídá počtu bitů, které jsou potřeba k zakódování daného racionálního čísla, vektoru resp. matice. Závisí samozřejmě na konkrétním výpočetním modelu a existují i jiné definice velikosti zápisu, ale liší se pouze o aditivní či multiplikativní konstantu. Tím pádem výsledky, které odvodíme, zůstávají v platnosti (s jinými koeficienty).

Věta 1.17 (Vlastnosti velikosti zápisu)**.**

(1) $\sigma(z) = 2 + \lceil \log_2(|z| + 1)\rceil$, $z \in \mathbb{Z}$,

(2) $\sigma(a + b) \le \sigma(a) + \sigma(b)$, $a, b \in \mathbb{Z}$,

(3) $\sigma(ab) \le \sigma(a) + \sigma(b)$, $a, b \in \mathbb{Q}$,

(4) $\sigma(a/b) \le \sigma(a) + \sigma(b)$, $a, b \in \mathbb{Q}$, $b \neq 0$,

(5) $a \le 2^{\sigma(a)}$, $a \in \mathbb{Q}$. □

Věta 1.18. *Buď $A \in \mathbb{Z}^{n\times n}$. Pak $\sigma(\det(A)) < \sigma(A)$.*

Důkaz. Pro $n = 1$ platí tvrzení triviálně, tak předpokládejme $n \ge 2$. Za použití odhadu

$$\begin{aligned}|\det(A)| &= |\textstyle\sum_{p\in S_n} \operatorname{sgn}(p) a_{1,p(1)} \cdots a_{n,p(n)}|\\ &\le \textstyle\sum_{p\in S_n} |a_{1,p(1)}| \cdots |a_{n,p(n)}| \le \prod_{i,j=1}^n (|a_{ij}| + 1)\end{aligned}$$

a věty 1.17(1) odvodíme

$$\begin{aligned}\sigma(\det(A)) &\le 2 + \left\lceil \log_2(1 + \textstyle\prod_{i,j=1}^n (|a_{ij}| + 1))\right\rceil \le 2 + \left\lceil \log_2(2\prod_{i,j=1}^n (|a_{ij}| + 1))\right\rceil\\ &\le 3 + \left\lceil \textstyle\sum_{i,j=1}^n \log_2(|a_{ij}| + 1)\right\rceil \le 3 + \textstyle\sum_{i,j=1}^n \lceil \log_2(|a_{ij}| + 1)\rceil\\ &< n^2 + \textstyle\sum_{i,j=1}^n (2 + \lceil \log_2(|a_{ij}| + 1)\rceil) = \sigma(A).\end{aligned}$$ □

Věta 1.19.

(1) Buď $A \in \mathbb{Z}^{n\times n}$ regulární a buď $b \in \mathbb{Z}^n$. Pak řešení x^* soustavy $Ax = b$ splňuje

$$\sigma(x_i^*) < 2\sigma(A \,|\, b),\ \forall i.$$

(2) Buď $A \in \mathbb{Z}^{m\times n}$ hodnosti m, a buď $m < n$. Pak soustava $Ax = 0$ má nenulové řešení x^* splňující

$$\sigma(x_i^*) < 2\sigma(A),\ \forall i.$$

Důkaz.

(1) Podle Cramerova pravidla je $x_i^* = \det(A_i^b)/\det(A)$, kde matice A_i^b vznikne z A nahražením i-tého sloupce vektorem b. Podle věty 1.17(4) a věty 1.18 je $\sigma(x_i^*) \le \sigma(\det(A_i^b)) + \sigma(\det(A)) < \sigma(A_i^b) + \sigma(A) \le 2\sigma(A \,|\, b)$.

(2) Soustavu $Ax = 0$ můžeme po vhodné permutaci proměnných vyjádřit jako $A_1x_1 + A_2x_2 = 0$, kde A_1 je regulární. Všechna řešení jsou tvaru $x_1 = -A_1^{-1}A_2x_2$, kde $x_2 \in \mathbb{R}^{n-m}$ je libovolné. Zvolme $x_2^* := e_1$. Pak x_1^* je řešením soustavy $A_1x_1 = -(A_2)_{*1}$, takže podle první části věty mají všechny složky velikost menší než $2\sigma(A)$. □

Poznámka 1.20. Nadále budeme bez újmy na obecnost předpokládat, že v úloze celočíselného programování (1.1) jsou A, b celočíselné.

Důkaz. Buď i libovolné pevné a φ velikost i-tého řádku matice $(A\,|\,b)$. Nejmenší společný násobek jmenovatelů čísel $a_{i1}, \ldots, a_{in}, b_i$ je nanejvýš 2^φ, neboť označíme-li tyto jmenovatele $q_1, \ldots, q_{n+1}$, tak podle věty 1.17(5)

$$q_1 \cdots q_{n+1} \leq 2^{\sigma(q_1)+\ldots+\sigma(q_{n+1})} \leq 2^{\sigma(a_{i1})+\ldots+\sigma(a_{in})+\sigma(b_i)} \leq 2^\varphi.$$

Tudíž přenásobením nejmenším společným násobkem se velikost jmenovatelů nezvětší a velikost každého čitatele zvětší maximálně o φ. Velikost i-tého řádku se tak zvětší maximálně $(n+2)$-krát. Tím pádem i velikost celé soustavy se zvětší maximálně $(n+2)$-krát. Podobné pro vektor c. □

Nyní připomeneme známou Carathéodoryho větu o tom, že konvexní obal v prostoru $\mathbb{R}^n$ lze zkonstruovat konvexními kombinacemi nanejvýš $n+1$ bodů. Druhá část věty se týká konvexního kužele, generovaného množinou S a definovaného jako

$$\operatorname{conv}\operatorname{cone}(S) = \left\{ \textstyle\sum_{i=1}^k \alpha_i x_i;\ x_i \in S,\ \alpha_i \geq 0,\ k \in \mathbb{N} \right\}.$$

Je to tedy nejmenší konvexní kužel obsahující množinu S.

Věta 1.21 (Carathéodory, 1911)**.**

(1) Je-li $S \subseteq \mathbb{R}^n$, $x \in \operatorname{conv}(S)$, pak $x \in \operatorname{conv}\{x_0, \ldots, x_n\}$ pro určité $x_0, \ldots, x_n \in S$.

(2) Je-li $S \subseteq \mathbb{R}^n$, $x \in \operatorname{conv}\operatorname{cone}(S)$, pak $x \in \operatorname{conv}\operatorname{cone}\{x_1, \ldots, x_n\}$ pro určité $x_1, \ldots, x_n \in S$.

Důkaz.

(1) Nechť $x = \sum_{i=0}^m \alpha_i x_i$, kde $\alpha_i > 0$, $\sum_i \alpha_i = 1$. Pokud je $m \leq n$, jsme hotovi. Jinak uvažme $x_1 - x_0$, $x_2 - x_0$, …, $x_m - x_0$. Těchto vektorů je m v n-dimenzionálním prostoru, tudíž jsou lineárně závislé a existují $\beta_1, \ldots, \beta_m$ takové, že alespoň jedna β_i je nenulová a $\sum_{i=1}^m \beta_i(x_i - x_0) = 0$. Položme $\beta_0 = -\sum_{i=1}^m \beta_i$. Pak $\sum_{i=0}^m \beta_i x_i = 0$ a $\sum_{i=0}^m \beta_i = 0$. Proto platí

$$x = \sum_{i=0}^m \alpha_i x_i = \sum_{i=0}^m \alpha_i x_i - \lambda \sum_{i=0}^m \beta_i x_i = \sum_{i=0}^m (\alpha_i - \lambda\beta_i) x_i \quad \forall \lambda \in \mathbb{R}.$$

Nyní můžeme zvolit λ tak, aby $\alpha_i' = \alpha_i - \lambda\beta_i$ byla všechna nezáporná a alespoň jedno nulové. Pak stále platí $\sum_{i=0}^m \alpha_i' = 1$, ale α_i' nenulových je méně než nenulových α_i. Takto můžeme dále pokračovat, dokud je vektorů x_i více než $n+1$.

(2) Dokáže se obdobně jako (1). □

Věta 1.22. *Buď $A \in \mathbb{Z}^{m\times n}$, $b \in \mathbb{Z}^m$ a označme φ největší velikost řádku matice $(A\,|\,b)$. Existuje-li celočíselný bod v $M = \{Ax \leq b,\ x \geq 0\}$, pak existuje celočíselný bod x^* takový, že $\sigma(x_i^*) < \Phi$ pro každé $i = 1, \ldots, m$, kde*

$$\Phi := 9n^3\varphi. \tag{1.4}$$

Důkaz. Podobné jako ve větě 1.1 můžeme každý bod $x \in M \cap \mathbb{Z}^n$ vyjádřit jako

$$x = \textstyle\sum_{i\in I} \alpha_i v_i + \sum_{j\in J} \beta_j h_j,$$

kde v_i, $i \in I$, jsou vrcholy M; h_j, $j \in J$, jsou směry neomezených hran M; $\alpha_i \geq 0$, $\sum_{i\in I} \alpha_i = 1$, $\beta_j \geq 0$. Navíc předpokládejme, že h_j jsou celočíselné.

Podle Carathéodoryho věty 1.21 lze předpokládat, že maximálně $n+1$ z čísel α_i je nenulových a maximálně n z čísel β_i je nenulových. Tedy bez újmy na obecnosti nechť $|I| = n+1$, $|J| = n$.

Podle věty 1.19 je velikost složek vektorů v_i a h_j nanejvýš $2n\varphi$. Protože jsme ale předpokládali, že h_j jsou celočíselné, tak vynásobením nejmenším společným násobkem jmenovatelů se může velikost zvýšit nanejvýš o $2n^2\varphi$ (srov. Poznámka 1.20). Tudíž velikost složek vektorů v_i a h_j odhadneme shora hodnotou $4n^2\varphi$. Pro bod $x^* \in M \cap \mathbb{Z}^n$ definovaný

$$x^* \coloneqq \textstyle\sum_{i\in I} \alpha_i v_i + \sum_{j\in J} \{\beta_j\} h_j$$

pak podle věty 1.17(5) platí

$$|x^*_k| \leq \textstyle\sum_{i\in I} |(v_i)_k| + \sum_{j\in J} |(h_j)_k| < (2n+1)2^{4n^2\varphi} = 2^{\log_2(2n+1)4n^2\varphi} \leq 2^{8n^3\varphi}.$$

Tím pádem podle věty 1.17(1)

$$\sigma(x^*_k) = 2 + \lceil \log_2(|x^*_k| + 1) \rceil \leq 2 + \lceil 8n^3\varphi \rceil < 9n^3\varphi. \qquad \square$$

Poznamenejme, že odhady byly dost hrubé a je možno dosáhnout těsnějších odhadů s menšími koeficienty. Pro naše účely to však postačuje.

Věta 1.23. *Problém rozhodnout zda $M \cap \mathbb{Z}^n \neq \emptyset$ je NP-úplný.*

Důkaz. Podle věty 1.15 je problém NP-těžký a podle věty 1.22 náleží do třídy NP, neboť máme garanci existence certifikátu polynomiální velikosti (to je onen celočíselný bod). $\square$

Poznámka 1.24 (Neomezenost M_I)**.** Bez újmy na obecnosti můžeme nadále předpokládat, že M_I je omezený. Podle věty 1.1 a jejího důkazu platí $M_I = Q_I + C$, tudíž všechny vrcholy M_I se vyskytují v Q_I. Pokud tedy optimální řešení x^* úlohy (1.1) existuje, tak podle důkazu věty 1.22 splňuje $|x^*_i| \leq 2^\Phi$. Případnou neomezenost lze proto testovat takto:

Řešme omezenou úlohu celočíselného programování

$$\max\ c^T x \ \text{ za podm. } \ x \in M \cap \mathbb{Z}^n,\ |x_i| \leq 2^\Phi\ \forall i.$$

Je-li nepřípustná, pak je i původní úloha. Je-li x^* optimální řešení a $|x^*_i| = 2^\Phi$ pro nějaké i, tak je původní úloha neomezená. Jinak je původní úloha omezená a x^* je její optimální řešení. Viz obrázek 1.4.

Poznámka 1.25. Celočíselný program není v NP, protože to není rozhodovací úloha. Odpovídající rozhodovací problém má tvar

$$M \cap \{x;\, c^T x \geq c_0\} \cap \mathbb{Z}^n \neq \emptyset, \tag{1.5}$$

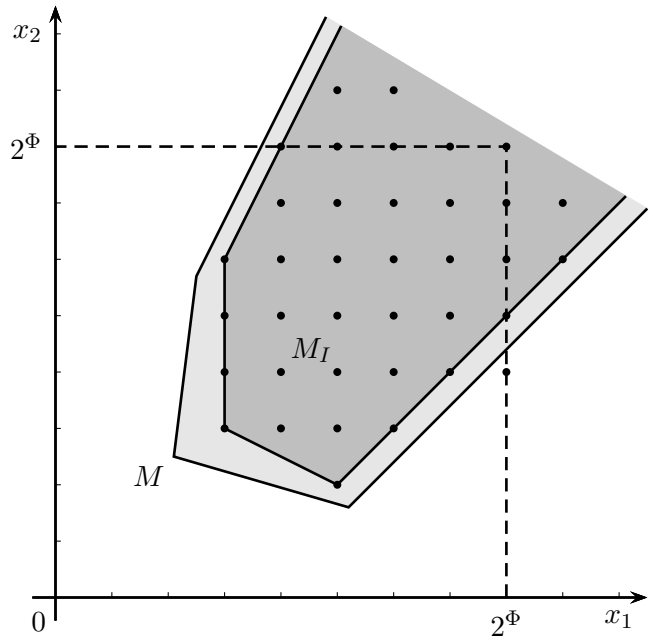
a vlastně se ptá, jestli existuje přípustné řešení s hodnotou účelové funkce aspoň c_0. Tento problém už NP-úplný jest.

Vztah úloh (1.1) a (1.5) je úzký – kdybychom dokázali rozhodnout (1.5) v polynomiálním čase, tak dokážeme řešit rovněž celočíselné programování v polynomiálním čase. Důvod je následující. Protože podle (1.4) jsou složky optimálního řešení v absolutní hodnotě menší než 2^Φ, tak (celočíselná) optimální hodnota je ve vnitřku intervalu $[-\alpha, \alpha]$, kde $\alpha \coloneqq n2^\Phi \sum_{i=1}^n |c_i|$. Tudíž pomocí binárního půlení parametru $c_0 \in [-\alpha, \alpha]$ a iterativním řešením rozhodovacích problémů (1.5) vyřešíme původní optimalizační úlohu po řádově $\log_2(2\alpha)$ iteracích, což je polynomiální hodnota vzhledem k velikosti vstupu.

Poznámka 1.26. Další NP-úplné problémy:

- Předchozí úlohy s polyedrem $M = \{x;\, Ax = b,\ x \geq 0\}$.
- Dáno $y \in \mathbb{Q}^n$ a M, patří $y \in M_I$?

Na druhou stranu, testování resp. nalezení celočíselného řešení soustavy rovnic $Ax = b$ již polynomiální problém je (tzv. diofantické rovnice). Další polynomiální případ je při pevné velikosti dimenze:



Obrázek 1.4: Neomezenost M_I, poznámka 1.24.

Věta 1.27 (Lenstra, 1983)**.**

(1) Pro každé $n \in \mathbb{N}$ pevné existuje polynomiální algoritmus, který rozhodne, zda polyedr $\{x \in \mathbb{R}^n;\ Ax \leq b\}$ obsahuje celočíselný bod, a pokud ho obsahuje, tak ho i najde.

(2) Pro každé $n \in \mathbb{N}$ pevné existuje polynomiální algoritmus, který řeší úlohu celočíselného programování $\max\{c^T x;\ Ax \leq b,\ x \in \mathbb{Z}^n\}$. □

Poznámka 1.28. Ne všechny celočíselné problémy jsou v třídě P či NP. Např. rozhodnout o celočíselné řešitelnosti polynomiální rovnice s celočíselnými koeficienty je nerozhodnutelný problém, známý jako 10. Hilbertův problém. Toto ukázal Matijasevič roku 1970.

Cvičení

1.3. Dokažte větu 1.17. Ukažte navíc, že vztah $\sigma(a+b) \leq \sigma(a) + \sigma(b)$ obecně neplatí pro $a, b \in \mathbb{Q}$.

1.5 Umění formulace celočíselných programů

Celočíselné programy mají velkou vyjadřovací schopnost. Pomocí nich můžeme zformulovat celou řadu netriviálních podmínek – logické či kvadratické podmínky, absolutní hodnotu, komplementaritu atp.

Příklad 1.29 (Min, max, absolutní hodnota aj.)**.** Zformulujte pomocí lineárních celočíselných podmínek následující úlohy:

(1) restrikce hodnot proměnných: $x \in \{v_1, \ldots, v_m\}$, kde $v_1, \ldots, v_m \in \mathbb{R}$ jsou dány,

(2) $u = \min(x, y)$, kde $0 \leq x \leq K$, $0 \leq y \leq K$ jsou proměnné,

(3) $u = |x|$, kde $0 \leq x \leq K$.

Řešení:

(1) Přidáme m binárních proměnných $\alpha_1, \ldots, \alpha_m$ a podmínku $x = \sum_{i=1}^m \alpha_i v_i$, kde $\sum_{i=1}^m \alpha_i = 1$.

(2) Opět potřebujeme, aby hodnoty proměnných x, y byly omezené nějakou konstantou K. Přidáme proměnnou u, binární proměnnou δ a podmínky:

$$\begin{aligned} u &\leq x, & u &\geq x - 2K\delta, \\ u &\leq y, & u &\geq y - 2K(1-\delta). \end{aligned}$$

(3) Přidáme binární proměnnou δ a podmínky:

$$\begin{aligned} u &\geq x, & u &\leq x + 2K\delta, \\ u &\geq -x, & u &\leq -x + 2K(1-\delta), \end{aligned}$$

nebo převodem na předchozí případ transformací $|x| = -\min(x, -x)$. □

Příklad 1.30 (Logické operace)**.** Uvažujme úlohu investice kapitálu c mezi 7 projektů, kde i-tý projekt má výnos c_i a náklady a_i. Zformulujte úlohu jako celočíselný program s následujícími dodatečnými omezeními:

(1) bez omezení,

(2) chci investovat aspoň do 2, ale nanejvýš do 5 projektů,

(3) do projektu 5 lze investovat právě tehdy, když do projektu 6,

(4) investuji-li do projektu 3, musím i do projektu 4,

(5) investuji-li do projektu 1, nesmím pak do projektu 2.

Řešení:

(1) $\max c^T x$ za podm. $a^T x \leq c,\ x \in \{0,1\}^7$.

(2) $2 \leq e^T x \leq 5$,

(3) $x_5 = x_6$,

(4) $x_4 \geq x_3$,

(5) $x_2 \leq 1 - x_1$. □

Příklad 1.31 (Po částech lineární funkce)**.** Zformulujte pomocí lineárních celočíselných podmínek po částech lineární funkci $f(x)$ na intervalu $[x_0, x_m]$, přičemž body zlomů v $x_0, \ldots, x_m \in \mathbb{R}$ a hodnoty ve zlomech jsou $a_0, \ldots, a_m \in \mathbb{R}$.

Řešení: Zavedeme binární proměnné $y_1, \ldots, y_m \in \{0,1\}$, kde $y_i = 1$ říká, že x se nachází v intervalu $[x_{i-1}, x_i]$ a $y_i = 0$ říká, že nikoli. Dále zavedeme spojité proměnné $\lambda_0, \ldots, \lambda_m$, pomocí níž vytvoříme odpovídající konvexní kombinace. Hledaná formulace pro $z = f(x)$ pak je

$$\begin{aligned} & x = \sum_{i=0}^{m} \lambda_i x_i, \quad z = \sum_{i=0}^{m} \lambda_i a_i, \quad \sum_{i=0}^{m} \lambda_i = 1, \\ & \lambda_0 \leq y_1, \quad \lambda_m \leq y_m, \\ & \lambda_i \leq y_{i-1} + y_i, \quad i = 1, \ldots, m-1, \\ & \lambda_1, \ldots, \lambda_m \geq 0, \\ & y_1, \ldots, y_m \in \{0,1\}. \end{aligned}$$

Pozoruhodné na této formulaci je, že relaxováním celočíselnosti proměnných $y_1, \ldots, y_m \in \{0,1\}$ na $y_1, \ldots, y_m \in [0,1]$ dostaneme nejlepší možnou lineární relaxaci. Výraz $(x, z) = \sum_{i=0}^{m} \lambda_i (x_i, a_i)$ pak totiž popisuje konvexní kombinaci bodů $(x_0, a_0), \ldots (x_m, a_m)$. Projekcí do podprostoru proměnných x, z tak dostaneme konvexní obal bodů $(x_0, a_0), \ldots (x_m, a_m)$, což je nejlepší možný výsledek – lineární podmínky popisují konvexní polyedr a projekcí dostaneme zase konvexní polyedr, tudíž těsnější objekt nelze očekávat. □

Příklad 1.32 (Sjednocení polyedrů)**.** Zformulujte pomocí lineárních celočíselných podmínek sjednocení m omezených konvexních polyedrů.

Řešení 1: Pro $i = 1, \dots, m$ nechť polyedr P_i je popsán soustavou $A^i x \leq b^i$. Pak jejich sjednocení $\cup_{i=1}^m P_i$ je popsáno

$$\begin{aligned} & A^i x^i \leq b^i y_i, \quad i = 1, \dots, m, \\ & x = \sum_{i=1}^m x^i, \quad \sum_{i=1}^m y_i = 1, \\ & x, x^1, \dots, x^m \in \mathbb{R}^n, \quad y_1, \dots, y_m \in \{0, 1\}. \end{aligned}$$

Zde se využívá faktu, že pro omezený polyedr nastane $Ax \leq 0$ právě tehdy, když $x = 0$. Binární proměnná y_i pak vyjadřuje, za x bude náležet do polyedru P_i. Přesně řečeno, $y_i = 1$ znamená, že ano, a $y_i = 0$ znamená, že nemusí.

Tato formulace má podobnou vlastnost s jakou jsme se setkali již v minulém Přikladu 1.31. Pokud relaxujeme binární proměnné na tvar $y_1, \dots, y_m \in [0, 1]$, soustava nabyde formy

$$\begin{aligned} & A^i x^i \leq b^i, \quad i = 1, \dots, m, \\ & x = \sum_{i=1}^m y_i x^i, \quad \sum_{i=1}^m y_i = 1, \\ & x^i \in \mathbb{R}^n, \quad y_1, \dots, y_m \in [0, 1], \end{aligned}$$

která vyjadřuje, že x je lineární kombinací bodů $x^i \in P_i$, čili $x \in \operatorname{conv}(\cup_{i=1}^m P_i)$.

Řešení 2: Sjednocení $\cup_{i=1}^m P_i$ lze popsat i jako

$$\begin{aligned} & A^i x \leq b^i + (1 - y_i) K e, \quad i = 1, \dots, m, \\ & \sum_{i=1}^m y_i = 1, \\ & x \in \mathbb{R}^n, \quad y_1, \dots, y_m \in \{0, 1\}. \end{aligned}$$

Zde K representuje dost velké číslo takové, aby $A^i x \leq b^i + Ke$ pro všechna $x \in \cup_{i=1}^m P_i$. Potom $y_i = 1$ říká, že x náleží do P_i a $y_i = 0$, že nemusí. Výhoda této formulace je, že využívá menší počet proměnných. Zatímco v první formulaci jsme měli $n(m + 1)$ spojitých a m binárních proměnných, nyní máme pouze n spojitých a m binárních proměnných. Na druhou stranu, druhá formulace nemusí být tak silná jako ta první. Kvalitu formulace lze zesílit co nejtěsnější volbou konstanty K, či ještě lépe vektor Ke nahradit co nejtěsnějším vektorem K^i pro $i = 1, \dots, m$.

Speciálně pro dva polyedry dostaneme formulaci

$$\begin{aligned} & A^1 x \leq b^1 + yKe, \\ & A^2 x \leq b^2 + (1 - y) Ke, \\ & x \in \mathbb{R}^n, \quad y \in \{0, 1\}. \end{aligned}$$

□

Příklad 1.33 (Pevné ceny)**.** Přeformujte celočíselným programem účelová funkci ve tvaru

$$f(x) = \begin{cases} c^T x + d & \text{je-li } x > 0, \\ 0 & \text{je-li } x = 0. \end{cases}$$

Takto vypadají často poplatky v dopravních úlohách – za převoz žádného zboží neplatíme nic, ale za převoz určitého zboží platíme úměrně dle množství zboží plus nějaké další fixní výdaje či paušál.

Řešení: Vyjádříme $f(x) = c^T x + dy$, kde $y \in \{0, 1\}$ je binární proměnná, která určuje, zda $x > 0$. Podmínku $y = 1 \Leftrightarrow x > 0$ vyjádříme jako $x \leq yK$, $x \geq yK^{-1}$, kde K je dost velká konstanta. Poslední podmínka $x \geq yK^{-1}$ se často může vynechat, protože v praktických úlohách bývá $d > 0$. □

Příklad 1.34 (Problém lineární komplementarity)**.** Přeformulujte jako celočíselný program úlohu lineární komplementarity, což je úloha přípustnosti systému bez účelové funkce

$$y = Mz + q, \quad y, sz \geq 0, \quad y^T z = 0,$$

s proměnnými $y, z \in \mathbb{R}^n$.

Řešení: Stačí přeformulovat podmínku komplementarity $y^T z = 0$. Tu rozepíšeme jako

$$y_i \leq \alpha b_i, \quad z_i \leq \alpha(1 - b_i), \quad i = 1, \ldots, n,$$

kde $\alpha > 0$ je dostatečně velké (jak víme, s polynomiální velikostí) a $b_i \in \{0, 1\}$ jsou binární proměnné. □

Příklad 1.35 (Asymetrie)**.** V řadě příkladů této sekce se ve formulaci vyskytují podmínky

$$\sum_{i=1}^{m} y_i = 1, \quad y \in \{0, 1\}^m.$$

Takovéto formulace pak působí potíže v algoritmech branch & bound, viz sekce 3.4. Proto je dobré přeformulovat podmínky jiným způsobem.

Uvažujme substituci $W \coloneqq Ly$, neboli $y \coloneqq L^{-1}w$, kde

$$L = \begin{pmatrix} 1 & \ldots & \ldots & 1 \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & 1 \end{pmatrix}, \quad L^{-1} = \begin{pmatrix} 1 & -1 & 0 & 0 \\ 0 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & -1 \\ 0 & \ldots & 0 & 1 \end{pmatrix}.$$

Tedy, $y_1 = w_1 - w_2, \ldots, y_{m-1} = w_{m-1} - w_m$, $y_m = w_m$. Pokud například proměnná x má representovat diskrétní výběr jedné z hodnot $a_1, \ldots, a_m$, pak tradiční formulaci

$$x = \sum_{i=1}^{m} a_i y_i, \quad \sum_{i=1}^{m} y_i = 1, \quad y \in \{0, 1\}^m$$

nahradíme formulací

$$x = \sum_{i=1}^{m-1} a_i(w_i - w_{i+1}) + a_m w_m, \quad 1 = w_1 \geq w_2 \geq \ldots \geq w_m, \quad w \in \{0, 1\}^m,$$

neboli

$$x = a_1 w_1 + \sum_{i=2}^{m} (a_i - a_{i-1} w_i, \quad 1 = w_1 \geq w_2 \geq \ldots \geq w_m, \quad w \in \{0, 1\}^m.$$

Cvičení

1.4. Buďte b_1, b_2, b_3 binární proměnné. Linearizujte podmínku $b_1 b_2 = b_3$.

1.5. Výroba produktu A nebo B implikují výrobu produktu C nebo D nebo E.

1.6. Zformulujte jako celočíselný program $u \in \{x \in \mathbb{Z}^n;\, Ax \leq b\} \setminus \{x^*\}$, kde $x^* \in \mathbb{R}^n$ je dáno a polyedr je omezený.

1.7. Zformulujte problém nalezení kružnice maximální váhy v grafu jako úlohu celočíselného programování. Váhy jsou dány na hranách a váha kružnice je součet vah jejích hran.

1.8. Formulace úloh z teorie grafů pomocí celočíselného programu. Použijte jak rozepsaný tvar, tak i vyjádření s maticí sousednosti nebo incidence.

(a) Určete velikost maximální kliky grafu.

(b) Najděte maximální nezávislou množinu grafu.

(c) Najděte vrcholové pokrytí grafu.

(d) Určete barevnost grafu.

(e) Najděte maximální párování v grafu.

(f) Najděte maximální řez v grafu.

1.9. Pro $n \in \mathbb{N}$ sestavte celočíselný program, který vyřeší problém umístění n dam na šachovnici $n \times n$ tak, aby se navzájem neohrožovaly.

1.10. Dána velká krabice s rozměry k_1, k_2, k_3 a n malých krabic s rozměry m_1^i, m_2^i, m_3^i, $i = 1, \ldots, n$. Napište celočíselný program na složení maximálního počtu malých krabic do velké. Podotýkáme, že krabice lze celé otáčet, ale nelze umisťovat šikmo.

1.11. Případová studie: Máme 1 letiště na pevnině a n vrtných plošin na moři, h vrtulníků a každý vrtulník má kapacitu osob c a rychlost v. Dále známe vzdálenosti d_{ij} mezi i-tou a j-tou plošinou a vzdálenosti d_{0j} mezi letištěm j-tou plošinou. Koncem každého měsíce musíme obměnit osoby pracující na plošinách; na i-té plošině je jich p_i.

Navrhněte řešení tohoto problému, tj. navrhněte formulaci, diskutujte její vlastnosti, popř. navrhněte vylepšení (a jaké údaje byste k vylepšení potřebovali).

Kapitola 2

Metody sečných nadrovin aneb řeznictví pana Gomoryho

Uvažujme úlohu celočíselného programování ve tvaru

$$\max\ c^T x\ \text{ za podm. }\ x \in M,\ x_i \in \mathbb{Z}\ \forall i \in C \subseteq \{1, \ldots, n\},$$

kde

$$M = \{x \in \mathbb{R}^n;\ Ax \leq b,\ x \geq 0\}.$$

Princip metod sečných nadrovin spočívá nejprve ve vyřešení lineární relaxace

$$\max\ c^T x\ \text{ za podm. }\ x \in M. \tag{2.1}$$

Pokud optimální řešení x^* splňuje podmínky celočíselnosti, jsme hotovi. Pokud ne, sestrojíme sečnou nadrovinu $r^T x = s$, která odřízne x^* od polyedru, ale žádné přípustné řešení neodřízne. To jest, musí platit

$$r^T x^* > s$$

a přitom

$$r^T x \leq s \text{ pro všechna } x \in M,\quad x_i \in \mathbb{Z}\ \forall i \in C.$$

Podmínku $r^T x \leq s$ přidáme do omezení M a celý postup opakujeme.

Pro řešení jednotlivých lineárních programů je vhodná tzv. *ℓ-metoda*, neboli lexikografická duální simplexová metoda. Duální simplexová metoda totiž umožňuje jednoduše přidávat dynamicky nové podmínky a rychle přeoptimalizovat. Lexikografie pak je užitečná pro konvergenci algoritmu. Následující tři sekce jsou proto věnované stručnému popisu této metody na řešení lineárních programů, a teprve v dalších sekcích se setkáme s algoritmy na úlohy celočíselného programování. Konkrétně, sekce 2.4 popisuje první Gomoryho algoritmus na čistou úlohu a sekce 2.5 druhý Gomoryho algoritmus na smíšenou úlohu celočíselného programování.

2.1 Duální simplexová metoda

Mějme matici $A \in \mathbb{R}^{m \times n}$ a primární úlohu lineárního programování ve tvaru:

$$\begin{aligned}\max_{x \in M_P}\ c^T x, \quad \text{kde } M &= \{x \in \mathbb{R}^n;\ Ax \leq b,\ x \geq 0\} \\ &= \{x \in \mathbb{R}^n;\ Ax + Ix' = b,\ x, x' \geq 0\}.\end{aligned}$$

K ní přísluší duální úloha

$$\begin{aligned}\min_{y \in M_D}\ b^T y, \quad \text{kde } M_D &= \{y \in \mathbb{R}^m;\ A^T y \geq c,\ y \geq 0\} \\ &= \{y \in \mathbb{R}^m;\ A^T y - Iy' = c,\ y, y' \geq 0\}.\end{aligned}$$

V revidované formě (bez jednotkové matice) je tabulka simplexové metody pro primární úlohu

	nebázické x	
bázické x'	A	b
	c^T	0

a pro duální úlohu

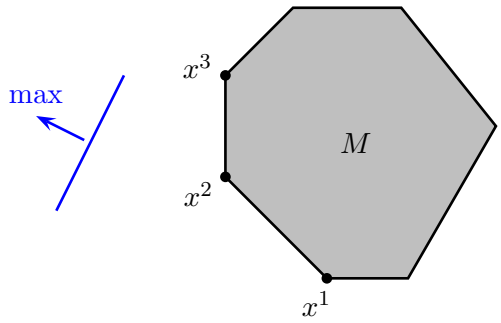
	nebázické y	
bázické y'	$-A^T$	$-c$
	b^T	0

Vidíme, že obě tabulky jsou v zásadě stejné až na překlopení a znaménko. Myšlenka duální simplexové metody je řešit duální úlohu za použití primární tabulky.

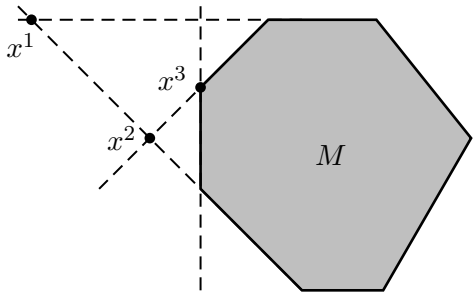
	nebáz. x, báz. y'	
báz. x', nebáz. y	A	b
	c^T	0

Zatímco primární simplexová metoda začíná s nezápornou pravou stranou b a postupnými iteracemi dospěje k výsledné tabulce s nezáporným kriteriálním řádkem $-c^T$, duální metoda to dělá naopak: začne s nezáporným kriteriálním řádkem a postupnými iteracemi dospěje k výsledné tabulce s nezápornou pravou stranou.

Z geometrického hlediska primární simplexová metoda prochází přípustná bázická řešení řešení, tedy vrcholy polyedru M, se vzrůstající hodnotou účelové funkce tak dlouho, dokud nenarazí na optimum. Duální simplexové metoda naopak prochází přípustná řešení duální úlohy, která odpovídají nepřípustným bázickým řešením primární úlohy. Postupně pak zlepšuje hodnotu účelové funkce duálu (a tedy zhoršuje hodnotu účelové funkce primáru), než získáme přípustné řešení primáru, které bude zároveň jeho optimem.



Primární simplexová metoda: postupná řešení x^1, x^2, x^3 jsou navazující vrcholy polyedru M.



Duální simplexová metoda: postupná řešení x^1, x^2, x^3 jsou nepřípustné báze.

2.2 Lexikografie

Definice 2.1. Buď $x, y \in \mathbb{R}^n$. Pak x je *lexikograficky větší* než y, psáno $x \succ y$, pokud $\exists i$ tak, že $x_1 = y_1, \ldots, x_{i-1} = y_{i-1}$ a $x_i > y_i$. Dále, x je *lexikograficky větší nebo rovno* y, psáno $x \succeq y$, pokud $x \succ y$ nebo $x = y$.

Lexikografie je tedy úplné uspořádání na $\mathbb{R}^n$, každé dva vektory jsou porovnatelné.

Pro $x \in M$ označme $\widetilde{x} \coloneqq (c^T x, x) \in \mathbb{R}^{n+1}$ jako *rozšířené řešení*. Dále řekneme, že x^* je *ℓ-optimální řešení* úlohy $\max_{x\in M} c^T x$ pokud $\widetilde{x}^* \succeq \widetilde{x}$ pro všechna $x \in M$. Je zřejmé, že ℓ-optimální řešení je zároveň optimální.

Věta 2.2. *Pro úlohu* $\max_{x\in M} c^T x$ *nechť je množina optimálních řešení* M_{opt} *neprázdná a omezená. Pak ℓ-optimální řešení existuje a je jednoznačné.*

Důkaz. Protože M_{opt} je omezený neprázdný konvexní polyedr, každý bod $x \in M_{\text{opt}}$ se dá vyjádřit jako konvexní kombinace $x = \sum_{i\in I} \alpha_i v_i$, kde v_i, $i \in I$, jsou vrcholy M_{opt} a $\alpha_i \geq 0$, $\sum_{i\in I} \alpha_i = 1$. Uvažme lexikograficky největší vrchol v_{i^*}, tj., $v_{i^*} \succeq v_i$ pro všechna $i \in I$. Nyní

$$x = \textstyle\sum_{i\in I} \alpha_i v_i \preceq \sum_{i\in I} \alpha_i v_{i^*} = v_{i^*},$$

tedy i pro bod $x \in M_{\text{opt}} \setminus \{x^*\}$ je $\widetilde{x} \prec \widetilde{v}_{i^*}$. Podobně pro libovolné $x \in M \setminus M_{\text{opt}}$ máme $c^T x < c^T v_{i^*}$, a proto $\widetilde{x} \prec \widetilde{v}_{i^*}$. Tudíž v_{i^*} je ℓ-optimální řešení. Jednoznačnost je zřejmá, neboť dva různé vektory jsou lexikograficky různé. □

2.3 ℓ-metoda

Jak jsme již zmínili, lineární programy (2.1) budeme řešit ℓ-metodou. Ta pracuje s ℓ-tabulkou, která má na začátku tvar

	x_1	x_n	
x_0	$-c^T$		0
x_1			0
$\vdots$	$-I_n$		$\vdots$
x_n			0
x_{n+1}			
$\vdots$	A		b
x_{m+n}			

$\equiv$

	x_{n_1}		x_{N_n}	
x_0				
x_1				
$\vdots$				
x_n	R_1	$\dots$	R_n	R_0
x_{n+1}				
$\vdots$				
x_{m+n}				

Na začátku jsou nebázické indexy $N \coloneqq \{1, \dots, n\}$. V průběhu algoritmu si tabulku budeme značit pomocí matice R, a sloupečky zleva jako $R_1, \dots, R_n, R_0$. Každý řádek tabulky odpovídá (ne)rovnici z omezení tvaru

$$x_i = R_{i0} - \sum_{j=1}^{n} R_{ij} x_{N_j}, \quad \forall i = 1, \dots, m+n.$$

U počáteční tabulky speciálně máme $x = 0 - (-x)$ a pro doplňkové proměnné $x' = b - Ax$, což souhlasí.

Úvodní krok

ℓ-tabulka je *ℓ-normální* pokud $R_j \succ 0$, $j = 1, \dots, n$ (nestačí tedy pouze nezápornost prvního řádku). Pokud je tabulka ℓ-normální, můžeme jít na běžný krok algoritmu, jinak musíme provést úvodní krok.

Do tabulky přidáme podmínku $\sum_{j=1}^{n} x_i \leq L$, kde $L > 0$ je dost velké. Hodnotu L buďto umíme vyčíst za zadání, nebo ji určíme pomocným lineárním programem, anebo vystupuje jako parametr reprezentující libovolně velkou hodnotu. Nová podmínka je přípustná pokud je úloha omezená; v opačném případě neomezenost odhalíme během algoritmu.

Klíčový řádek bude ten nový a klíčový sloupec ten lexikograficky nejmenší. Provedeme pivotizaci dle běžných simplexových pravidel tak, aby v klíčovém řádku byl opačný jednotkový vektor, tzn., že klíčový sloupec se dělí opačnou hodnotou pivota. Označme jako R' tabulku po jedné transformaci. Pak transformace s pivotem na pozici (k, ℓ) má tvar:

$$R'_\ell \coloneqq \frac{1}{-R_{k\ell}} R_\ell, \tag{2.2a}$$

$$R'_j \coloneqq R_j - \frac{R_{kj}}{R_{k\ell}} R_\ell, \quad j \neq \ell. \tag{2.2b}$$

Tvrzení 2.3. *Po jedné transformaci je tabulka ℓ-normální.*

Důkaz. Buď R_ℓ klíčový sloupec a označme R' tabulku po jedné transformaci. Protože $R_\ell \prec 0$, bude $R'_\ell = -R_\ell \succ 0$. Dále, z definice ℓ pro každé $j \neq \ell$ máme $R'_j = R_j - R_\ell \succ 0$. □

Běžný krok

Volba klíčového řádku k pro pivota:

$$k \coloneqq \arg\min\{i = 1, \ldots, m+n;\ R_{i0} < 0\}.$$

Volba klíčového sloupce ℓ pro pivota:

$$\frac{R_\ell}{|R_{k\ell}|} \coloneqq \operatorname{lex\,min}\left\{ \frac{R_j}{|R_{kj}|};\ j = 1, \ldots, n,\ R_{kj} < 0 \right\}.$$

Transformace tabulky probíhá podle pravidla (2.2).

Tvrzení 2.4. *Po transformaci tabulka zůstane ℓ-normální.*

Důkaz. Pro klíčový sloupec platí

$$R'_\ell = \frac{1}{-R_{k\ell}} R_\ell \succ 0.$$

Pro ostatní sloupce $j \neq \ell$ platí

$$R'_j = R_j - \frac{R_{kj}}{R_{k\ell}} R_\ell.$$

Pokud $R_{kj} \geq 0$, tak triviálně $R'_j \succ 0$. Pokud $R_{kj} < 0$, tak z definice ℓ je $\frac{1}{-R_{kj}} R_j \succ \frac{1}{-R_{k\ell}} R_\ell$. Odtud $R_j \succ \frac{-R_{kj}}{-R_{k\ell}} R_\ell$ a proto $R'_j \succ 0$. □

Tvrzení 2.5. *Poslední sloupec tabulky se po každé transformaci lexikograficky zmenší.*

Důkaz. Pro poslední sloupec tabulky platí stejná transformační pravidla. Protože $R_{k0}, R_{k\ell} < 0$, dostáváme

$$R'_0 = R_0 - \frac{R_{k0}}{R_{k\ell}} R_\ell \prec R_0.$$ □

Důsledek 2.6. *ℓ-metoda je konečná.*

Důkaz. Každá tabulka odpovídá jedné bázi, a těch je konečně mnoho. Protože se po každé transformaci poslední sloupec lexikograficky zmenší, algoritmus musí skončit po konečně mnoha iteracích. □

Tvrzení 2.7 (kriterium optimality)**.** *Je-li $x^* \coloneqq (R_{1,0}, \ldots, R_{m+n,0}) \geq 0$, pak x^* je ℓ-optimální řešení.*

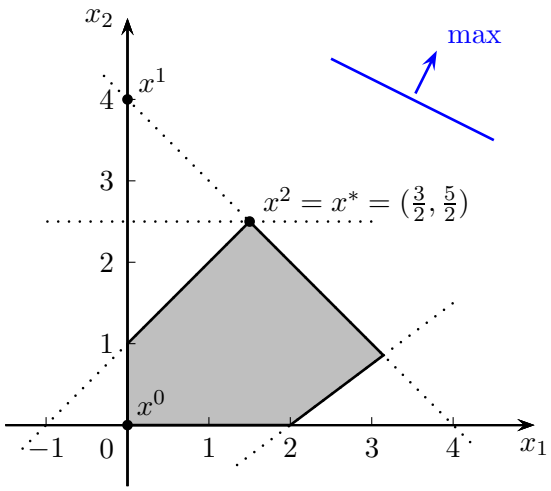
Důkaz. Optimalita x^* vyplývá z teorie duální simplexové metody.

Z důkazu tvrzení 2.2 víme, že ℓ-optimální řešení je vrcholem M, takže k důkazu ℓ-optimality x^* stačí ukázat, že přechodem k jinému optimálnímu bázickému řešení (což lze učinit konečně mnoha přípustnými transformacemi ℓ-tabulky) si pouze lexikograficky pohoršíme vektor $R_0 = \widetilde{x}^*$.

Uvažme transformaci dle pivota $R_{k\ell} \neq 0$. Předpokládejme, že $R_{k0} > 0$, neboť případ $R_{k0} = 0$ nevede ke změně vrcholu. Pokud $R_{k\ell} > 0$, tak $R'_0 = R_0 - \frac{R_{k0}}{R_{k\ell}} R_\ell \prec R_0$ a tedy si pohoršíme. Pro případ $R_{k\ell} < 0$ nechť $s \coloneqq N_\ell$. Pak s-tý řádek tabulky je $-e_\ell$ a tudíž $R'_{s0} = R_{s0} - \frac{R_{k0}}{R_{k\ell}} R_{s\ell} = 0 + \frac{R_{k0}}{R_{k\ell}} < 0$. To znamená, že by řešení nebylo přípustné. □

Tvrzení 2.8 (kriterium nepřípustnosti)**.** *Nechť pro index $k > 0$ platí: $R_{k0} < 0$ a $R_{kj} \geq 0$, $j = 1, \ldots, n$. Pak $M = \emptyset$.*

Důkaz. k-tý řádek ℓ-tabulky odpovídá rovnici $x_k = R_{k0} - \sum_{j=1}^{n} R_{kj} x_{N_j} < 0$, což nesplňuje žádný přípustný (a tedy nezáporný) vektor x. □



Obrázek 2.1: (Příklad 2.10) Průběh ℓ-metody.

Poznámka 2.9. Jak je vidět, v ℓ-metodě závisí na pořadí proměnných a omezení. Různou volbou pořadí na začátku tedy můžeme ovlivnit průběh celého algoritmu.

Příklad 2.10. Řešte lineární program

$$\max \; x_1 + 2x_2$$

za podmínek

$$\begin{aligned} -x_1 + x_2 &\le 1, & x_1 + 3x_2 &\le 9, \\ 3x_1 - 4x_2 &\le 6, & x_2 &\le \tfrac{5}{2}, \\ x_1 + x_2 &\le 4, & x &\ge 0. \end{aligned}$$

Řešení: Sestavíme tabulku a provádíme jednotlivé transformace. Pivot je zakroužkován a v záhlaví jsou nebázické proměnné. Protože první tabulka není ℓ-normální, provádíme Úvodní krok s tím, že použijeme nerovnosti $x_1 + x_2 \le 4$ ze zadání.

	x_1	x_2	
x_0	−1	−2	0
x_1	−1	0	0
x_2	0	−1	0
x_3	−1	1	1
x_4	3	−4	6
x_5	1	(1)	4
x_6	1	3	9
x_7	0	1	$\frac{5}{2}$

$\sim$

	x_1	x_5	
x_0	1	2	8
x_1	−1	0	0
x_2	1	1	4
x_3	(−2)	−1	−3
x_4	7	4	22
x_5	0	−1	0
x_6	−2	−3	−3
x_7	−1	−1	$-\frac{3}{2}$

$\sim$

	x_3	x_5	
x_0	$\frac{1}{2}$	$\frac{3}{2}$	$\frac{13}{2}$
x_1	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{3}{2}$
x_2	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{5}{2}$
x_3	−1	0	0
x_4	$\frac{7}{2}$	$\frac{1}{2}$	$\frac{23}{2}$
x_5	0	−1	0
x_6	−1	−2	0
x_7	$-\frac{1}{2}$	$-\frac{1}{2}$	0

Z výsledné tabulky vyčteme optimální řešení $(x_1^*, x_2^*) = (\frac{3}{2}, \frac{5}{2})$ a optimální hodnotu $\frac{13}{2}$. Dále vidíme, že řešení je silně degenerované, neboť doplňkové proměnné $x_3^* = x_5^* = x_6^* = x_7^* = 0$ (pro nedegenerované řešení by pouze dvě doplňkové proměnné byly nulové).

Postup algoritmu znázorňuje obrázek 2.1. Začínáme v bodě $x^0 = (0, 0)$ a postupujeme skrze bod $x^1 = (0, 4)$ do výsledného optimálního řešení $x^2 = x^* = (\frac{3}{2}, \frac{5}{2})$.

Pro ilustraci ještě ukážeme, jak by metoda pracovala, kdybychom nerovnost $x_1 + x_2 \le 4$ neuměli odvodit ze zadání. Použijeme nerovnost $x_1 + x_2 \le L$, kde L representuje libovolně velké číslo.

	x_1	x_2	
x_0	-1	-2	0
x_1	-1	0	0
x_2	0	-1	0
x_3	-1	1	1
x_4	3	-4	6
x_5	1	1	4
x_6	1	3	9
x_7	0	1	$\frac{5}{2}$
x_5	1	$\textcircled{1}$	L

$\sim$

	x_1	x_5	
x_0	1	2	$2L$
x_1	-1	0	0
x_2	1	1	L
x_3	$\textcircled{-2}$	-1	$1-L$
x_4	7	4	$6+4L$
x_5	0	-1	$4-L$
x_6	-2	-3	$9-3L$
x_7	-1	-1	$\frac{5}{2}-L$

$\sim$

	x_3	x_5	
x_0	$\frac{1}{2}$	$\frac{3}{2}$	$\frac{1}{2}+\frac{3}{2}L$
x_1	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}+\frac{1}{2}L$
x_2	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}+\frac{1}{2}L$
x_3	-1	0	0
x_4	$\frac{7}{2}$	$\frac{1}{2}$	$\frac{19}{2}+\frac{1}{2}L$
x_5	0	$\textcircled{-1}$	$4-L$
x_6	-1	-2	$8-2L$
x_7	$-\frac{1}{2}$	$-\frac{1}{2}$	$2-\frac{1}{2}L$

$\sim$

	x_3	x_5	
x_0	$\frac{1}{2}$	$\frac{3}{2}$	$\frac{13}{2}$
x_1	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{3}{2}$
x_2	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{5}{2}$
x_3	-1	0	0
x_4	$\frac{7}{2}$	$\frac{1}{2}$	$\frac{23}{2}$
x_5	0	-1	0
x_6	-1	-2	0
x_7	$-\frac{1}{2}$	$-\frac{1}{2}$	0

S proměnnou L pracujeme jako s parametrem, což prakticky lze implementovat tak, že sloupec pravých stran se skládá ze dvou podsloupců – první representuje absolutní členy a druhý koeficienty u L. □

2.4 První Gomoryho algoritmus

Uvažujme čistou úlohu celočíselného programování ve tvaru

$$\max\ c^T x \ \text{ za podm. } \ x \in M \cap \mathbb{Z}^n,$$

kde $A \in \mathbb{Z}^{m\times n}$, $b \in \mathbb{Z}^m$, $c \in \mathbb{Z}^n$ a

$$M = \{x \in \mathbb{R}^n;\ Ax \le b,\ x \ge 0\}.$$

Schema prvního Gomoryho algoritmu[1)] [Gomory, 1958] je následující:

Algoritmus 2.11 (První Gomoryho algoritmus (1958))**.**

1: Vyřešíme lineární program $\max_{x\in M} c^T x$ pomocí ℓ-metody. Pokud optimální řešení neexistuje, tak ani původní úloha nemá řešení. Pokud je optimální řešení celočíselné, končíme.

2: Jinak zavedeme řez následujícím způsobem. Definujme první řádek s neceločíselnými hodnotami napravo v tabulce jako

$$k \coloneqq \min\{i = 0,\dots,m+n;\ R_{i0} \notin \mathbb{Z}\}. \tag{2.3}$$

a nadrovinu

$$H \coloneqq \{x \in \mathbb{R}^n;\ \textstyle\sum_{j=1}^n \{R_{kj}\} x_{N_j} = \{R_{k0}\}\}.$$

K omezením z M přidáme další podmínku

$$x \in H^+ = \{x \in \mathbb{R}^n;\ \textstyle\sum_{j=1}^n \{R_{kj}\} x_{N_j} \ge \{R_{k0}\}\}$$

a celý postup opakujeme.

Nyní ukážeme, že nadrovina odřízne aktuální optimální řešení x^* lineárního programu, ale neodřízne žádný celočíselný bod v M.

Věta 2.12 (přípustnost řezu)**.** *Platí $x^* \in \operatorname{int} H^-$ a $M \cap \mathbb{Z}^n \subseteq H^+$, kde*

$$\operatorname{int} H^- = \{x \in \mathbb{R}^n;\ \textstyle\sum_{j=1}^n \{R_{kj}\} x_{N_j} < \{R_{k0}\}\},$$
$$H^+ = \{x \in \mathbb{R}^n;\ \textstyle\sum_{j=1}^n \{R_{kj}\} x_{N_j} \ge \{R_{k0}\}\}.$$

[1)] Gomory původně bádal ve své disertaci na nelineárních diferenciálních rovnicích. K celočíselnému programování se dostal když pracoval pak pro americkou armádu.

Důkaz. 1) Vlastnost $x^* \in \operatorname{int} H^-$ plyne dosazením x^* do nerovnice $\sum_{j=1}^n \{R_{kj}\} x^*_{N_j} < \{R_{k0}\}$, neboť $x^*_N = 0$.

2) Buď $x \in M \cap \mathbb{Z}^n$ libovolný. Pak

$$x_k = R_{k0} - \sum_{j=1}^n R_{kj} x_{N_j} = \lfloor R_{k0} \rfloor - \sum_{j=1}^n \lfloor R_{kj} \rfloor x_{N_j} + \{R_{k0}\} - \sum_{j=1}^n \{R_{kj}\} x_{N_j}.$$

Z této rovnice vyplývá, že $\sum_{j=1}^n \{R_{kj}\} x_{N_j} - \{R_{k0}\} \in \mathbb{Z}$. Protože ale

$$\sum_{j=1}^n \underbrace{\{R_{kj}\}}_{\geq 0} \underbrace{x_{N_j}}_{\geq 0} - \underbrace{\{R_{k0}\}}_{<1} > -1,$$

musí $\sum_{j=1}^n \{R_{kj}\} x_{N_j} - \{R_{k0}\} \geq 0$. □

Poznámka 2.13.

1. V algoritmu může nastat i situace $k = 0$, což je v pořádku.
2. Je-li $\{R_{kj}\} = 0$ pro každé $j \in \{1, \ldots, n\}$, pak $M \cap \mathbb{Z}^n = \emptyset$. Tedy i když nadrovina není dobře definována, věta stále platí.
3. Přidání omezení $x \in H^+$ prakticky znamená, že do ℓ-tabulky přidáme další řádek

$\vdots$		$\vdots$		$\vdots$
x_{m+n+1}	$-\{R_{k1}\}$	$\ldots$	$-\{R_{kn}\}$	$-\{R_{k0}\}$

   odpovídající omezení $x_{m+n+1} = -\{R_{k0}\} - \sum_{j=1}^n -\{R_{ij}\} x_{N_j}$. Pokračujeme ℓ-metodou s tím, že klíčový řádek bude ten poslední.
4. Řez je jednoduchý, ale nepříliš hluboký.

Tvrzení 2.14. *Po provedení řezu a jedné transformace tabulky lze přidaný řádek vynechat.*

Důkaz. Sloupce R_j, $j = 1, \ldots, n$, tabulky zůstanou lineárně nezávislé, a proto jednoznačnost výběru klíčového sloupce přetrvá. Původní omezení z popisu polyedru M rovněž zůstanou zachovány, tudíž se zachová přípustnost řešení. Nevrátíme se ani zpět k již spočítanému řešení, protože ℓ-tabulka lexikograficky klesá. □

Nyní směřujeme k tomu ukázat, že První Gomoryho algoritmus je konečný. Označme jako R_0^r poslední sloupec výsledné ℓ-tabulky (tj. rozšířené ℓ-optimální řešení) úlohy v r-té iteraci

$$\max\ c^T x \ \text{ za podm. } \ x \in M \cap H_1^+ \cap \cdots \cap H_r^+.$$

Nechť R'^r_0 značí poslední sloupec po provedení řezu a jedné transformace tabulky.

Lemma 2.15. *Platí* $R_0^r \succ R'^r_0 \succeq R_0^{r+1}$.

Důkaz. Plyne z tvrzení 2.5. Rovnost napravo nastane pokud po jedné transformaci tabulky je už R'^r_0 přímo ℓ-optimální. □

Lemma 2.16. *Existuje vektor* $v \in \mathbb{Z}^{m+n+1}$ *tak, že* $R_0^r \geq v$ *pro každé* $r = 0, 1, \ldots$.

Důkaz. Stačí volit $v = (\alpha, 0, \ldots, 0)^T$, kde α je celočíselná dolní mez na optimální hodnotu. □

Lemma 2.17. *Buď* p *libovolné takové, že* $R^r_{p0} \notin \mathbb{Z}$. *Pak*

$$(R^r_{00}, \ldots, R^r_{p-1,0}, \lfloor R^r_{p,0} \rfloor) \succeq (R'^r_{00}, \ldots, R'^r_{p-1,0}, R'^r_{p,0}). \tag{2.4}$$

Důkaz. Pro jednoduchost vynecháme horní index r a budeme psát R namísto R^r. Řez konstruujeme podle k-tého řádku tabulky a podle (2.3) je $k \leq p$. Definujme

$$q := \min\{i = 0, \ldots, m+n;\ R_{i\ell} \neq 0\}$$

jako index první nenulové hodnoty v ℓ-tém sloupci. Protože $R_{k\ell} \neq 0$, máme $q \leq k \leq p$.

	$R_\ell \succ 0$	R_0
	0	
q	$R_{q\ell} > 0$	
k	$R_{k\ell} \neq 0$	$R_{k0} \notin \mathbb{Z}$
p		$R_{p0} \notin \mathbb{Z}$
	$-\{R_{k\ell}\}$	$-\{R_{k0}\} < 0$

Rozlišme dva případy:

(a) Nechť $q < p$. Ze vztahu pro pravý sloupec tabulky po jedné transformaci $R'_0 = R_0 - \frac{-\{R_{k0}\}}{-\{R_{k\ell}\}} R_\ell$ dostáváme speciálně pro q-tou složku (předchozí hodnoty se nezmění)

$$R'_{q0} = R_{q0} - \frac{-\{R_{k0}\}}{-\{R_{k\ell}\}} R_{q\ell} < R_{q0}.$$

Tudíž $(R_{00}, \ldots, R_{q0}) \succ (R'_{00}, \ldots, R'_{q0})$ a proto i (2.4).

(b) Nechť $q = k = p$. Ze vztahu pro k-tou složku pravého sloupce tabulky po jedné transformaci dostáváme

$$R'_{k0} = R_{k0} - \frac{-\{R_{k0}\}}{-\{R_{k\ell}\}} R_{k\ell} = R_{k0} - \frac{R_{k\ell}}{\{R_{k\ell}\}} \{R_{k0}\} \leq R_{k0} - \{R_{k0}\} = \lfloor R_{k0} \rfloor,$$

kde jsme využili $R_{k\ell} \geq \{R_{k\ell}\}$. Nyní tedy dostáváme (2.4).

□

Věta 2.18. *První Gomoryho algoritmus je konečný.*

Důkaz. Sporem předpokládejme, že algoritmus je nekonečný a vytvoří posloupnost postupných řešení R^r, $r = 0, 1, \ldots$ Podle Lemmatu 2.15 je $R^0 \succ R^2 \succ \ldots$ Tudíž dostáváme nerostoucí posloupnost

$$R^0_{00} \geq R^1_{00} \geq R^2_{00} \geq \ldots$$

Podle Lemmatu 2.16 je posloupnost zdola omezená. Může existovat index r_0 takový, že posloupnost je konstantní od r_0-tého členu. Pak je nerostoucí posloupnost

$$R^{r_0}_{10} \geq R^{r_0+1}_{10} \geq R^{r_0+2}_{10} \geq \ldots$$

Obecně, uvažujme minimální $s \in \{0, \ldots, m+n\}$ takové, že posloupnost R^0_{s0}, R^1_{s0}, $R^2_{s0}, \ldots$ se nikde neustálí. Pak od určitého členu je posloupnost nerostoucí, zdola omezená a proto má limitu. Z konvergence posloupnosti tedy existuje $z \in \mathbb{Z}$ takové, že od určitého členu jsou prvky posloupnosti v intervalu $(z, z+1)$. Formálně, existuje r_s takové, že pro všechna $r \geq r_s$ je $R^r_{s0} \in (z, z+1)$. Nyní použijeme Lemma 2.17 pro $p := s$, což je přípustné díky $R^r_{s0} \notin \mathbb{Z}$. Z lemmatu vyplývá, že

$$(R^r_{00}, \ldots, R^r_{s-1,0}, \lfloor R^r_{s,0} \rfloor) \succeq (R'^r_{00}, \ldots, R'^r_{s-1,0}, R'^r_{s0}),$$

neboli $R^{r+1}_{s0} \leq R'^r_{s0} \leq \lfloor R^r_{s0} \rfloor = z$, což je spor. □

Poznámka 2.19. Pro důkaz konečnosti algoritmu jsme potřebovali konkrétní znalost řezu pouze v Lemmatu 2.17, část (b). Proto pro důkaz konečnosti jiných algoritmů sečných nadrovin stačí pouze adaptovat tuto část důkazu.

Příklad 2.20. Řešte celočíselný lineární program

$$\max\ x_1 - x_2$$

za podmínek

$$-\tfrac{1}{3}x_1 + x_2 \le \tfrac{1}{3}, \quad x_1 - \tfrac{1}{3}x_2 \le \tfrac{1}{3}, \quad x \ge 0,\ x \in \mathbb{Z}^2.$$

Řešení: Omezení přepíšeme do tvaru s celočíselnými koeficienty

$$-x_1 + 3x_2 \le 1, \quad 3x_1 - x_2 \le 1, \quad x \ge 0,\ x \in \mathbb{Z}^2.$$

Z nerovnic odvodíme vztah $x_1 + x_2 \le 1$, který použijeme pro úvodní krok ℓ-metody. Sestavíme ℓ-tabulku a vyřešíme lineární relaxaci ℓ-metodou.

	x_1	x_2	
x_0	-1	1	0
x_1	-1	0	0
x_2	0	-1	0
x_3	-1	3	1
x_4	3	-1	1
x_5	(1)	1	1

$\sim$

	x_5	x_2	
x_0	1	2	1
x_1	1	1	1
x_2	0	-1	0
x_3	1	4	2
x_4	(-3)	-4	-2

$\sim$

	x_4	x_2	
x_0	$\frac{1}{3}$	$\frac{2}{3}$	$\frac{1}{3}$
x_1	$\frac{1}{3}$	$-\frac{1}{3}$	$\frac{1}{3}$
x_2	0	-1	0
x_3	$\frac{1}{3}$	$\frac{8}{3}$	$\frac{4}{3}$
x_4	-1	0	0

Protože řešení $(\frac{1}{3}, 0)$ relaxované úlohy nesplňuje celočíselné podmínky, zavedeme řez. Volíme $k = 0$ a řezná nadrovina bude $\frac{1}{3}x_4 + \frac{2}{3}x_2 = \frac{1}{3}$. Přidáme novou podmínku do tabulky a přeoptimalizujeme. Jak vidíme dole, stačí jedna transformace. Protože řešení ještě stále nebude celočíselné, zavedeme druhý řez (pro $k = 1$) a po transformaci tabulky už získáme celočíselné optimální řešení.

	x_4	x_2	
x_0	$\frac{1}{3}$	$\frac{2}{3}$	$\frac{1}{3}$
x_1	$\frac{1}{3}$	$-\frac{1}{3}$	$\frac{1}{3}$
x_2	0	-1	0
x_3	$\frac{1}{3}$	$\frac{8}{3}$	$\frac{4}{3}$
x_4	-1	0	0
x_6	$-\frac{1}{3}$	$(-\frac{2}{3})$	$-\frac{1}{3}$

$\sim$

	x_4	x_6	
x_0	0	1	0
x_1	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$
x_2	$\frac{1}{2}$	$-\frac{3}{2}$	$\frac{1}{2}$
x_3	-1	4	0
x_4	-1	0	0
x_7	$(-\frac{1}{2})$	$-\frac{1}{2}$	$-\frac{1}{2}$

$\sim$

	x_7	x_6	
x_0	0	1	0
x_1	1	-1	0
x_2	1	-2	0
x_3	-2	5	1
x_4	-2	1	1

Z výsledné tabulky vyčteme optimální celočíselné řešení $(x_1^*, x_2^*) = (0, 0)$ a optimální hodnotu 0.

Je užitečné se podrobněji podívat na sečné nadroviny. První měla tvar $\frac{1}{3}x_4 + \frac{2}{3}x_2 = \frac{1}{3}$. Za proměnnou x_4 dosadíme ze čtvrtého řádku úplně první tabulky předpis $x_4 = 1 - 3x_1 + x_2$, čímž dostáváme tvar nadroviny $x_1 - x_2 = 0$ v proměnných x_1, x_2. Podobně druhá sečná nadrovina měla tvar $\frac{1}{2}x_4 + \frac{1}{2}x_6 = \frac{1}{2}$. Dosazením $x_6 = -\frac{1}{3} + \frac{1}{3}x_4 + \frac{2}{3}x_2$ a $x_4 = 1 - 3x_1 + x_2$ dostaneme tvar $2x_1 - x_2 = 0$ sečné nadroviny.

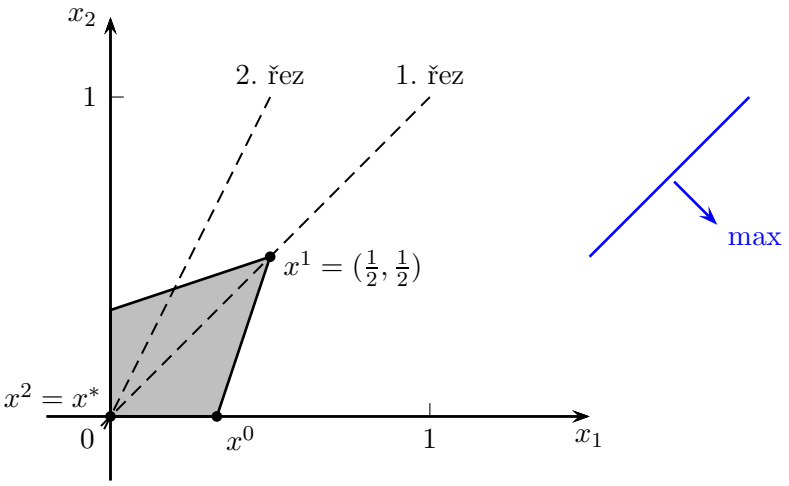
Postup algoritmu a sečné nadroviny jsou znázorněny na obrázku 2.2. Optimum relaxované úlohy je $x^0 = (\frac{1}{3}, 0)$. Po prvním řezu dospějeme k řešení $x^1 = (\frac{1}{2}, \frac{1}{2})$ a po druhém řezu dojdeme konečně k celočíselnému optimu $x^2 = x^* = (0, 0)$.

Poznamenejme, že kdybychom na začátku nepřeváděli nerovnosti na tvar s celočíselnými koeficienty, tak by algoritmus nepracoval korektně a vrátil chybné řešení (zkuste to!). □

2.5 Druhý Gomoryho algoritmus

Uvažujme smíšenou úlohu celočíselného programování ve tvaru

$$\max\ c^T x \ \text{ za podm. } \ x \in M_C,$$



Obrázek 2.2: (Příklad 2.20) Průběh prvního Gomoryho algoritmu.

kde $A \in \mathbb{Q}^{m\times n}$, $b \in \mathbb{Q}^m$, $c \in \mathbb{Q}^n$, $C \subseteq \{1, \ldots, n\}$ a

$$M = \{x \in \mathbb{R}^n;\ Ax \leq b,\ x \geq 0\},$$
$$M_C = \{x \in M;\ x_i \in \mathbb{Z} \text{ pro } i \in C\}.$$

Pro důkaz konečnosti se navíc předpokládá, že $C = \{1, \ldots, n_0\}$ pro nějaké $n_0 \leq n$ a $x_0 = c^T x \in \mathbb{Z}$ (druhá podmínka už není na újmu obecnosti).

Schema druhého Gomoryho algoritmu [Gomory, 1960] je následující:

Algoritmus 2.21 (Druhý Gomoryho algoritmus (1960))**.**

1: Vyřešíme lineární program $\max_{x\in M} c^T x$ pomocí ℓ-metody. Pokud optimální řešení neexistuje, tak ani původní úloha nemá řešení.

Pokud optimální řešení splňuje celočíselné podmínky, končíme.

2: Jinak zavedeme řez následujícím způsobem. Definujme

$$k \coloneqq \arg\min\{i \in C;\ R_{i0} \notin \mathbb{Z}\}.$$

a nadrovinu

$$H \coloneqq \{x \in \mathbb{R}^n;\ \textstyle\sum_{j=1}^n \gamma_j x_{N_j} = \{R_{k0}\}\},$$

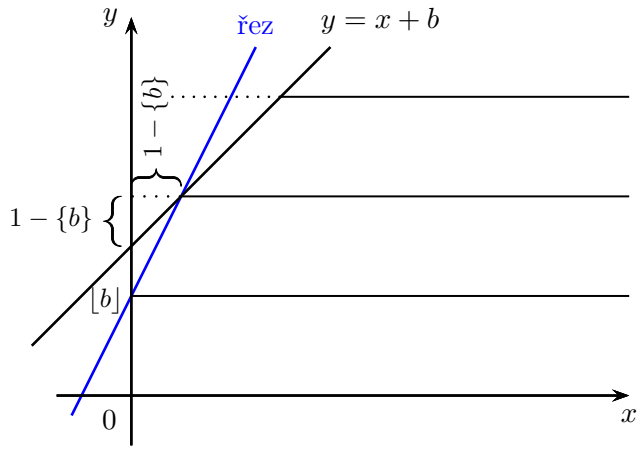
kde γ_j, $j \in N$, jsou definovány

$$\gamma_j = \begin{cases} \{R_{kj}\} & N_j \in C,\ \{R_{k0}\} \geq \{R_{kj}\}, \quad (2.5a) \\ \dfrac{\{R_{k0}\}}{1-\{R_{k0}\}}(1-\{R_{kj}\}) & N_j \in C,\ \{R_{k0}\} < \{R_{kj}\}, \quad (2.5b) \\ R_{kj} & N_j \notin C,\ R_{kj} \geq 0, \quad (2.5c) \\ \dfrac{\{R_{k0}\}}{1-\{R_{k0}\}}(-R_{kj}) & N_j \notin C,\ R_{kj} < 0. \quad (2.5d) \end{cases}$$

K omezením z M přidáme další podmínku

$$x \in H^+ = \{x \in \mathbb{R}^n;\ \textstyle\sum_{j=1}^n \gamma_j x_{N_j} \geq \{R_{k0}\}\}$$

a celý postup opakujeme.



Obrázek 2.3: (Věta 2.22) Podstata řezné přímky.

Nadrovina odřízne aktuální optimální řešení x^* lineárního programu, ale neodřízne žádný přípustný bod v M_C. Nejprve dokážeme pomocné lemma.

Lemma 2.22. *Uvažujme omezení*

$$y - x \leq b, \quad x, y \geq 0, \quad x \in \mathbb{R}, \ y \in \mathbb{Z},$$

kde $b \notin \mathbb{Z}$. Pak následující řezná nerovnost je přípustná

$$y - \frac{1}{1-\{b\}} x \leq \lfloor b \rfloor.$$

Důkaz. Neformálně je vidět platnost řezu na obrázku 2.3. Řezná přímka musí procházet bodem $(0, \lfloor b \rfloor)$ a mít směrnici $\frac{1}{1-\{b\}}$. Pro formální důkaz rozlišme dva případy:

1) Je-li $y \leq \lfloor b \rfloor$, pak je tvrzení jasné, neboť ve vyjádření nerovnice jako $y - \lfloor b \rfloor \leq \frac{1}{1-\{b\}} x$ je levá strana nekladná a pravá nezáporná.

2) Nechť $y > \lfloor b \rfloor$, to jest, $y \geq \lfloor b \rfloor + 1 = \lceil b \rceil$. Nyní upravme

$$\begin{aligned} y - b &\leq x, \\ y - \lceil b \rceil + (1 - \{b\}) &\leq x, \\ (1 - \{b\})(y - \lceil b \rceil) + (1 - \{b\}) &\leq x, \\ (1 - \{b\})(y - \lfloor b \rfloor) &\leq x, \end{aligned}$$

z čehož vyplývá výsledná řezná nerovnost. □

Tvrzení 2.23 (Přípustnost Druhého Gomoryho řezu)**.** *Platí $x^* \in \operatorname{int} H^-$ a $M_C \subseteq H^+$, kde*

$$\begin{aligned} H^- &= \{x \in \mathbb{R}^n; \textstyle\sum_{j=1}^n \gamma_j x_{N_j} < \{R_{k0}\}\}, \\ H^+ &= \{x \in \mathbb{R}^n; \textstyle\sum_{j=1}^n \gamma_j x_{N_j} \geq \{R_{k0}\}\}. \end{aligned}$$

Důkaz. Vlastnost $x^* \in \operatorname{int} H^-$ plyne dosazením x^* do nerovnice $\sum_{j=1}^n \gamma_j x^*_{N_j} < \{R_{k0}\}$, neboť $x^*_N = 0$.

Nyní se zaměříme na vlastnost $M_C \subseteq H^+$. Buď $x \in M_C$ libovolný pevný. Podle k-tého řádku tabulky platí

$$x_k = R_{k0} - \textstyle\sum_{j=1}^n R_{kj} x_{N_j}.$$

Rovnici přepíšeme jako dvě nerovnice

$$x_k + \textstyle\sum_{j=1}^n R_{kj} x_{N_j} \leq R_{k0}, \tag{2.6}$$

$$-x_k - \textstyle\sum_{j=1}^n R_{kj} x_{N_j} \leq -R_{k0}. \tag{2.7}$$

První nerovnici upravíme na

$$x_k + \sum_{(2.5a)} R_{kj}x_{N_j} + \sum_{(2.5b)} R_{kj}x_{N_j} + \sum_{(2.5c)} R_{kj}x_{N_j} + \sum_{(2.5d)} R_{kj}x_{N_j} \leq R_{k0}$$

a dále na

$$x_k + \sum_{(2.5a)} \lfloor R_{kj}\rfloor x_{N_j} + \sum_{(2.5b)} \lceil R_{kj}\rceil x_{N_j} + \sum_{(2.5c)} 0 \leq R_{k0} + \sum_{(2.5b)} (1-\{R_{kj}\})x_{N_j} - \sum_{(2.5d)} R_{kj}x_{N_j}.$$

Nyní použijeme Lemma 2.22, proměnné na pravé straně nerovnosti mají nezáporné koeficienty, proto můžeme lemma použít. Tím odvodíme platnou nerovnost

$$x_k + \sum_{(2.5a)} \lfloor R_{kj}\rfloor x_{N_j} + \sum_{(2.5b)} \lceil R_{kj}\rceil x_{N_j} + \sum_{(2.5c)} 0 \leq \lfloor R_{k0}\rfloor + \sum_{(2.5b)} \frac{1-\{R_{kj}\}}{1-\{R_{k0}\}}x_{N_j} - \sum_{(2.5d)} \frac{R_{kj}}{1-\{R_{k0}\}}x_{N_j}.$$

K této nerovnici přičteme nerovnici (2.7) a dostaneme

$$\sum_{(2.5a)} -\{R_{kj}\}x_{N_j} + \sum_{(2.5b)} ?x_{N_j} + \sum_{(2.5c)} -R_{kj}x_{N_j} + \sum_{(2.5d)} ?x_{N_j} \leq -\{R_{k0}\}.$$

Koeficient v sumě odpovídající (2.5b) je

$$\lceil R_{kj}\rceil - R_{kj} - \frac{1-\{R_{kj}\}}{1-\{R_{k0}\}} = 1-\{R_{kj}\} - \frac{1-\{R_{kj}\}}{1-\{R_{k0}\}} = \frac{-\{R_{k0}\}}{1-\{R_{k0}\}}(1-\{R_{kj}\}).$$

Koeficient v sumě odpovídající (2.5d) je

$$\frac{R_{kj}}{1-\{R_{k0}\}} - R_{kj} = \frac{\{R_{k0}\}}{1-\{R_{k0}\}}R_{kj}.$$

Odvodili jste tedy platnou nerovnost $\sum_{j=1}^{n} -\gamma_j x_{N_j} \leq -\{R_{k0}\}$. □

Poznámka 2.24.

1. Po provedení řezu a jedné transformace tabulky lze přidaný řádek vynechat.
2. Řez je konstruován tak, aby fungoval pro smíšenou úlohu a aby byl co nejhlubší. Přesto hloubka řezu není nijak závratná.
3. Je-li $\gamma_j = 0$ pro všechna $j \in \{1,\ldots,n\}$, pak $M_C = \emptyset$, tedy řez je přípustný, i když nadrovina sama o sobě není dobře definována.
4. Přidání omezení $x \in H^+$ prakticky znamená, že do ℓ-tabulky přidáme další řádek

$\vdots$		$\vdots$		$\vdots$
x_{m+n+1}	$-\gamma_1$	$\ldots$	$-\gamma_n$	$-\{R_{k0}\}$

   odpovídající omezení $x_{m+n+1} = -\{R_{k0}\} - \sum_{j=1}^{n}(-\gamma_j)x_{N_j}$. Pokračujeme ℓ-metodou s tím, že klíčový řádek bude ten poslední.

Příklad 2.25. Řešte příklad 2.20 pomocí druhého Gomoryho algoritmu.

Řešení: Uvažovaná úloha má tvar

$$\max\ x_1 - x_2 \ \text{ za podm. } \ -\tfrac{1}{3}x_1 + x_2 \leq \tfrac{1}{3}, \quad x_1 - \tfrac{1}{3}x_2 \leq \tfrac{1}{3}, \quad x \geq 0,\ x \in \mathbb{Z}^2.$$

Omezení nemusíme přepisovat do tvaru s celočíselnými koeficienty; ve stávajícím tvaru máme $C = \{0,1,2\}$. Dále využijeme toho, že z nerovnic vyplývá vztah $x_1 + x_2 \leq 1$, který použijeme pro úvodní krok ℓ-metody. Sestavíme ℓ-tabulku a vyřešíme lineární relaxaci ℓ-metodou.

	x_1	x_2	
x_0	-1	1	0
x_1	-1	0	0
x_2	0	-1	0
x_3	$-\frac{1}{3}$	1	$\frac{1}{3}$
x_4	1	$-\frac{1}{3}$	$\frac{1}{3}$
x_5	$\textcircled{1}$	1	1

$\sim$

	x_5	x_2	
x_0	1	2	1
x_1	1	1	1
x_2	0	-1	0
x_3	$\frac{1}{3}$	$\frac{4}{3}$	$\frac{2}{3}$
x_4	$\textcircled{-1}$	$-\frac{4}{3}$	$-\frac{2}{3}$

$\sim$

	x_4	x_2	
x_0	1	$\frac{2}{3}$	$\frac{1}{3}$
x_1	1	$-\frac{1}{3}$	$\frac{1}{3}$
x_2	0	-1	0
x_3	$\frac{1}{3}$	$\frac{8}{9}$	$\frac{4}{9}$
x_4	-1	0	0

Protože řešení $(\frac{1}{3}, 0)$ relaxované úlohy nesplňuje celočíselné podmínky, zavedeme řez. Volíme $k = 0$ a řezná nerovnice má tvar $\sum_{j=1}^{n} \gamma_j x_{n_j} \leq -\{R_{k0}\} \leq -\frac{1}{3}$. Výpočet γ_j pro $j = 1, 2$:

1. Protože $N_1 = 4 \notin C$ a $R_{k1} = R_{01} = 1 \geq 0$, nastane třetí případ z definice γ_j a položíme $\gamma_1 = 1$.
2. Protože $N_2 = 2 \in C$ a $\frac{1}{3} = R_{k0} < R_{k2} = \frac{2}{3}$, nastane druhý případ z definice γ_j a položíme $\gamma_2 = \frac{1}{6}$.

Přidáme novou podmínku do tabulky a přeoptimalizujeme.

	x_4	x_2	
x_0	1	$\frac{2}{3}$	$\frac{1}{3}$
x_1	1	$-\frac{1}{3}$	$\frac{1}{3}$
x_2	0	-1	0
x_3	$\frac{1}{3}$	$\frac{8}{9}$	$\frac{4}{9}$
x_4	-1	0	0
x_6	$\textcircled{-1}$	$-\frac{1}{6}$	$-\frac{1}{3}$

$\sim$

	x_6	x_2	
x_0	1	$\frac{1}{2}$	0
x_1	1	$-\frac{1}{2}$	0
x_2	0	-1	0
x_3	$\frac{1}{3}$	$\frac{5}{6}$	$\frac{1}{3}$
x_4	-1	$\frac{1}{6}$	$\frac{1}{3}$

Jak vidíme, stačí jedna transformace a dostaneme optimální celočíselné řešení $(x_1^*, x_2^*) = (0, 0)$ a optimální hodnotu 0.

Jak konkrétně vypadá řezná nadrovina? Podle předpisu je tvaru $-x_4 - \frac{1}{6}x_2 = -\frac{1}{3}$. Proměnnou x_4 vyjádříme z první tabulky jako $x_4 = \frac{1}{3} - x_1 + \frac{1}{3}x_2$ a dosazením dostáváme popis sečné nadroviny v původních proměnných jako $-\frac{1}{3} + x_1 - \frac{1}{3}x_2 - \frac{1}{6}x_2 = -\frac{1}{3}$, neboli $x_1 - \frac{1}{2}x_2 = 0$. Je to tedy stejný řez, který První Gomoryho algoritmus sestrojil až v druhé iteraci. □

Poznámka 2.26. Historii Gomoryho řezů provází ve vědecké komunitě oscilace mezi přijetím a odmítáním. Nejprve byly považovány za zázračné, pak zcela odmítnuty ve prospěch metod branch & bound (kapitola 3) a nakonec opět povolány do boje s celočíselnými programy. Klíčovou novou myšlenkou, kromě řady jiných, bylo aplikovat více řezů najednou (viz sekce 3.3); najít vhodnou kombinaci řezů je však stále otevřený problém. Úspěšná byla i kombinovace s jinými přístupy a také se celkem podařilo zvládnout zásadní problémy se zaokrouhlováním v tabulce.

2.6 Další řezy

2.6.1 Integer rounding a Chvátalovy–Gomoryho řezy

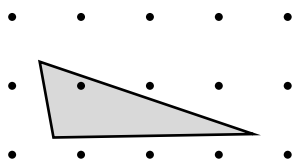
Příklad 2.27. Uvažme

$$M = \{x \in \mathbb{Z}^4;\ \tfrac{13}{11}x_1 + \tfrac{20}{11}x_2 + x_3 + \tfrac{6}{11}x_4 \geq \tfrac{72}{11}\}.$$

Zaokrouhlením koeficientů vlevo nahoru, a poté pravé strany rovněž nahoru dostaneme platnou nerovnost (splněnou pro každý bod z M)

$$2x_1 + 2x_2 + x_3 + x_4 \geq 7.$$

Tato nerovnost odřízne např. bod $(0, 0, \frac{72}{11}, 0)$, který ta původní nerovnost splňuje. Pro lineární relaxaci má tedy smysl generovat takovéto řezy. □



Obrázek 2.4: Ilustrace příkladu, kdy ani všechny Chvátalovy–Gomoryho řezy nezmenší M na M_I, a tedy Chvátalův rank je větší než 1. Pravý vrchol trojúhelníku totiž neodřízneme více než o zhruba zlomkovou část a nedostaneme se mimo okolní celočíselné body.

Postup z příkladu můžeme zobecnit. Ve shodě se sekcí 1.2 uvažme polyedr

$$M = \{x \in \mathbb{R}^n;\ Ax \leq b,\ x \geq 0\}$$

a konvexní obal celočíselných bodů

$$M_I \coloneqq \operatorname{conv}(M \cap \mathbb{Z}^n).$$

Pak pro libovolné $u \geq 0$ je $u^T Ax \leq u^T b$ platná nerovnost pro M, a

$$\lfloor u^T A \rfloor x \leq \lfloor u^T b \rfloor$$

je platná nerovnost pro M_I, neodřízne tedy žádný celočíselný bod z M. Tato nerovnice se nazývá Chvátalův–Gomoryho řez. Přidáním všech takovýchto nerovnic pro všechna možná $u \geq 0$ zmenšíme M na nějaký menší polyedr (není vidno na první pohled, že je to polyedr). Nemusíme ještě dostat M_I (viz obrázek 2.4), ale pokud celý postup opakujeme, po konečně mnoha iteracích (tzv. Chvátalův rank) dospějeme k M_I. Toto tvrdí slavná Chvátalova věta z r. 1973, a platí i pro reálná data.

Příklad 2.28.

1. Najděte Chvátalův–Gomoryho řez, který odřízne bod $(0, \frac{25}{6}, 0, 0, 0)$ od polyedru

$$M = \{x \in \mathbb{Z}^5;\ 9x_1 + 12x_2 + 8x_3 + 17x_4 + 13x_5 \geq 50,\ x \geq 0\}.$$

2. Najděte Chvátalův–Gomoryho řez, který odřízne bod $(20, 5, \frac{25}{8})$ od polyedru

$$M = \{(x, y) \in \mathbb{R}^2 \times \mathbb{Z};\ x_1 + x_2 \leq 25,\ x_1 + x_2 \leq 8y,\ (x, y) \geq 0\}.$$

□

2.6.2 Generování logických nerovností

V 0-1 programování lze uvažovat i opačným směrem – generovat nové jednoduché omezení. Smysl těchto omezení je stejný jako u sečných nadrovin, nevyrábět redundantní nerovnice, ale omezení, která potenciálně odříznou část relaxovaného polyedru. Postup se nejlépe vysvětlí na příkladu.

Příklad 2.29. Uvažme celočíselný systém

$$\begin{aligned} 7x_1 + 3x_2 - 4x_3 - 2x_4 &\leq 1, \\ -2x_1 + 7x_2 + 3x_3 + x_4 &\leq 6, \\ -2x_2 - 3x_3 - 6x_4 &\leq -5, \\ -3x_1 + 2x_3 &\leq 1, \\ x &\in \{0,1\}^4. \end{aligned}$$

Z první nerovnice můžeme odvodit podmínky:

- Je-li $x_1 = 1$, pak $x_3 = 1$ (neboli $x_1 \leq x_3$).

- Je-li $x_1 = 1$, pak $x_4 = 1$ (neboli $x_1 \leq x_4$).
- Nemohou obě x_1, x_2 být zároveň rovny 1 (neboli $x_1 + x_2 \leq 1$).

Z druhé nerovnice podobně odvodíme podmínky:

- $x_1 \geq x_2$, $x_2 + x_3 \leq 1$.

Z třetí nerovnice odvodíme podmínky:

- $x_2 + x_4 \geq 1$, $x_3 + x_4 \leq 1$.

Z poslední nerovnice pak odvodíme podmínku:

- $x_1 \geq x_3$.

Zkombinováním podmínek dostaneme:

- $x_1 \geq x_3$, $x_1 \leq x_3$, tedy $x_1 = x_3$.
- $x_1 + x_2 \leq 1$, $x_1 \geq x_2$, tedy $x_2 = 0$.
- $x_2 + x_4 \geq 1$, $x_2 = 0$, tedy $x_4 = 1$.

Tudíž v zásadě bez výpočtu jsme odvodili $x_1 = x_3$, $x_2 = 0$, $x_4 = 1$, což skutečně dává všechna přípustná řešení $(1, 0, 1, 1)$ a $(0, 0, 0, 1)$. □

2.6.3 Perfektní párování a květinové nerovnosti

Edmondsův algoritmus [Edmonds, 1965] na nalezení perfektního párování je založen na lineárním programování, dualitě a sečných nadrovinách. Sečné nadroviny generujeme z faktu, že z množiny vrcholů liché velikosti musí vycházet aspoň jedna hrana. Navíc se dá ukázat, že všechny takovéto nerovnosti stačí k tomu, abychom původní relaxovaný polyedr ořízli na konvexní obal perfektních párování. S pomocí duality pak získáme polynomiální algoritmus.

Problém maximálního párování v grafu $G = (V, E)$ má formulaci

$$\max \sum_{e \in E} x_e \quad \text{za podm.} \quad Ax \leq e,\ x \in \{0,1\}^m,$$

kde $x_e = 1$ odpovídá hraně v párování a $x_e = o$ jinak. Platná sečná nadrovna pak pro libovolnou množinu vrcholů $T \subseteq V$ liché velikosti má tvar $\sum_{e \in E(T)} x_e \leq \frac{1}{2}(|T| - 1)$, kde $E(T)$ jsou hrany grafu G s oběma vrcholy uvnitř T. I když je těchto nerovnic exponenciálně mnoho, chytrým výběrem lze dostat polynomiální algoritmus.

Cvičení

2.1. Vyřešte příklad 2.20 pomocí prvního a druhého Gomoryho algoritmu a *primární* simplexové metody.

2.2. V prvním Gomoryho algoritmu po zavedení sečné nadroviny a provedení transformace tabulky podle nového řádku pak tento řádek vynecháme. Může se stát, že se někdy později dostaneme zase na druhou stranu této sečné nadroviny? Obecně, jak ovlivní vynechání řádku průběh algoritmu?

2.3. Adaptujete první Gomoryho algoritmus tak, abychom se vyhnuli řezům účelové funkce (tj. $k > 0$). Jak to bude s konečností? Dokážeme se jednoduše vyhnout i řezům pro doplňkové proměnné?

2.4. Zobecněte první Gomoryho algoritmus pro smíšenou úlohu. Vodítkem vám může být druhý Gomoryho algoritmus a důkaz správnosti sečné nadroviny.

2.5. Pro druhý Gomoryho algoritmus dokažte konečnost (za vhodných předpokladů). Stačí analogii Lemmatu 2.17 z prvního Gomoryho algoritmu.

Dále, diskutujte konečnost pro případ, že účelová funkce nemusí nabývat celočíselných hodnot.

2.6. Ovlivní nějak průběh prvního resp. druhého Gomoryho algoritmu to, že soustavu nerovnic předem přenásobím nějakým kladným (a např. celým) číslem?

2.7. Dokažte, že sečnou nadrovinu čisté úlohy celočíselného programu lze definovat např. také následujícím způsobem: Buď $d_{k0} \notin \mathbb{Z}$, tj. výsledná l-tabulka nemá napravo celé číslo v k-tém řádku. Sečná nadrovina je pak tvaru $\sum_{j \in I} x_{N_J} = 1$, kde $I = \{j = 1, \ldots, n;\, d_{kj} \notin \mathbb{Z}\}$ (množina indexů prvků v k-tém řádku, které nejsou celočíselné).

2.8. Buď $f > 0$ dáno. Najděte minimální popis konvexního obalu množiny

$$M = \{(x, y) \in \mathbb{Z} \times \mathbb{R};\, x - y \leq f,\ y \geq 0\}$$

pomocí soustavy lineárních nerovnic.

Kapitola 3

Metody branch & bound

Myšlenka metody branch & bound („větvení a mezí“) byla nezávisle dotknuta několika autory během 50. let 20. století. První úplný algoritmus by formulován v legendárním článku Land and Doig [1960][1)], který myšlenku proslavil, ale první rozumná implementace se objevuje až v Dakin [1965]. Název „branch & bound“ pochází od Little et al. [1963], kteří jej vymysleli pro rekurzivní metodu na řešení problému obchodního cestujícího.

Uvažujme smíšenou úlohu celočíselného programování ve tvaru

$$\max\ c^T x \ \text{ za podm. } \ x \in M_C,$$

kde $A \in \mathbb{Q}^{m\times n}$, $b \in \mathbb{Q}^m$, $c \in \mathbb{Q}^n$, $C \subseteq \{1,\dots,n\}$ a

$$M = \{x \in \mathbb{R}^n;\ Ax \le b,\ x \ge 0\},$$
$$M_C = \{x \in M;\ x_i \in \mathbb{Z} \text{ pro všechna } i \in C\}.$$

Schema metod založených na branch & bound je následující, viz obrázek 3.1.

Algoritmus 3.1 (Branch & bound)**.**

1: Vyřešíme lineární program $\max_{x\in M} c^T x$.

Pokud optimální řešení neexistuje, tak ani původní úloha nemá řešení.

Pokud je optimální řešení $x^0 \in M_C$, končíme.

2: Jinak zvol $k \in C$ takové, že $x^0_k \notin \mathbb{Z}$ a definuj dva konvexní polyedry

$$M_1 := \{x \in M;\ x_k \le \lfloor x^0_k \rfloor\},$$
$$M_2 := \{x \in M;\ x_k \ge \lfloor x^0_k \rfloor + 1\}.$$

Rekurzivně vyřeš dvě podúlohy na polyedrech M_1 resp. M_2 a vyber lepší řešení.

Pro řešení lineárních relaxací je opět výhodné použít duální simplexovou metodu (např. ℓ-metodu), protože potřebujeme rychle přeoptimalizovat po přidání nové podmínky $x_k \le \lfloor x^0_k \rfloor$ resp. $x_k \ge \lfloor x^0_k \rfloor + 1$. Pro $k \in C$ takové, že $x^0_k = R_{k0} \notin \mathbb{Z}$, k-tý řádek ℓ-tabulky má tvar

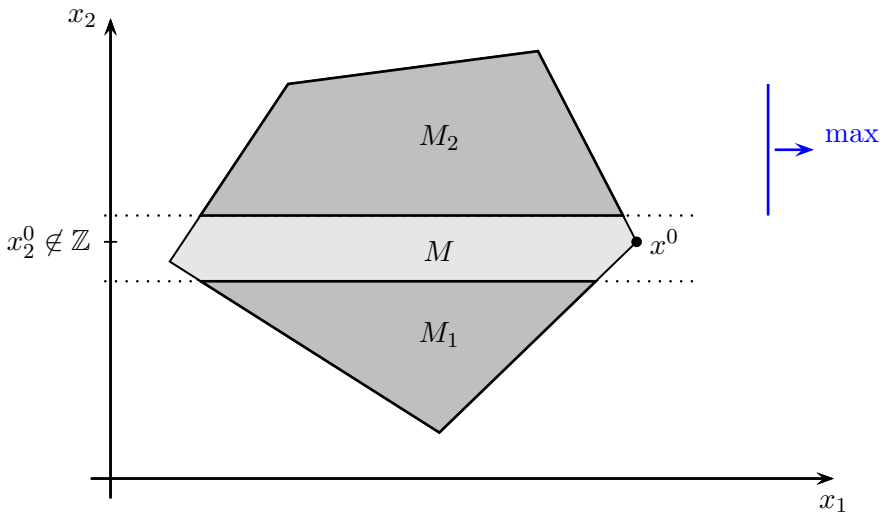
$$x_k = R_{k0} - \sum_{j=1}^{n} R_{kj} x_{N_j}.$$

Podmínka $x_k \le \lfloor x^0_k \rfloor$ pak vede na novou nerovnici

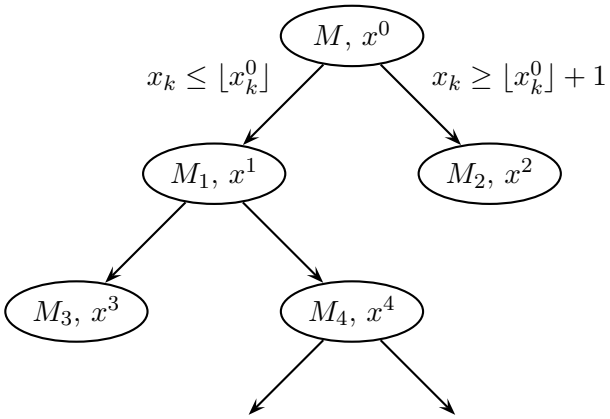
$$-\sum_{j=1}^{n} R_{kj} x_{N_j} \le -\{R_{k0}\}, \tag{3.1}$$

a podobně podmínka $x_k \ge \lfloor x^0_k \rfloor + 1$ vede na novou nerovnici

$$\sum_{j=1}^{n} R_{kj} x_{N_j} \le \{R_{k0}\} - 1.$$



Obrázek 3.1: Myšlenka metody branch & bound.



Obrázek 3.2: Strom representující volání podúloh metody branch & bound.

Čili do ℓ-tabulky přidáme koeficienty jedné z těchto nerovnic.

Poznámka 3.2 (Meze)**.** Algoritmus 3.1 popisuje schema větvení. Metoda tedy prochází binární strom representující volání podúloh, viz obrázek 3.2. Pro účinnou implementaci, abychom neprocházeli zbytečně všechny uzly stromu, potřebujeme efektivně zohlednit také „meze“. Nejjednodušším způsobem je pamatovat si aktuálně nejlepší přípustné řešení $x^* \in M_C$ (jakmile nějaké objevíme) a příslušnou hodnotu účelové funkce $z^* := c^T x^*$, která dává globální *dolní* mez na optimální hodnotu f^*. Naopak, horní mez na optimální hodnotu podúlohy dává příslušná lineární relaxace. Proto můžeme zavést test:

> Je-li optimální hodnota lineární relaxace dané podúlohy menší nebo rovna z^*, pak podúlohu nemusím řešit (podstrom ve výpočetním stromu odřízneme).

Poznámka 3.3 (Základní vlastnosti)**.**

- Jednoduše řeší smíšené úlohy, zejména je-li málo celočíselných proměnných.
- Je-li M omezený (srov. Poznámka 1.24), pak algoritmus je konečný. Každá proměnná totiž nabývá konečně mnoho hodnot a tudíž se podle ní mohu větvit konečně mnohokrát.
- Dá se snadno paralelizovat.

[1)] Ailsa Landová a Alison Doigová, matematičky z Londýnské školy ekonomie.

- Narozdíl od sečných nadrovin není tak citlivý na zaokrouhlovací chyby.
- Dá se jednoduše použít i pro nelineární úlohy celočíselného programování.
- Na druhou stranu, může být hodně pomalé a náročné na paměť.

Poznámka 3.4 (0-1 programování)**.** Pro 0-1 programování (sekce 1.1) mají podúlohy tvar

$$M_1 \coloneqq \{x \in M;\, x_k = 0\},$$
$$M_2 \coloneqq \{x \in M;\, x_k = 1\}.$$

S každou hladinou zanoření ve výpočetním stromu se tedy zafixuje jedna proměnná a dimenze úlohy (= počet proměnných) klesá.

Příklad 3.5. Tento příklad ukazuje, že metoda branch & bound může být časově náročná (exponenciální) i pro jednoduchou úlohu. Pro n liché uvaž

$$\max \sum_{i=1}^{n} 2x_i \;\text{ za podm. }\; \sum_{i=1}^{n} 2x_i \le n,\; x \in \{0,1\}^n.$$

Vidíme, že optimální řešení je např. prvních $\frac{1}{2}(n-1)$ proměnných rovno jedné, a ostatní nula; optimální hodnota je $n-1$. Na druhou stranu, lineární relaxace dá optimální hodnotu n pro každý uzel stromu až do hladiny $\frac{1}{2}(n-1)$. Musíme tudíž projít strom až do hloubky $\frac{1}{2}(n+1)$ a navštívit $2^{\frac{1}{2}(n+1)}$ uzlů. □

Větvení jde uvažovat ještě v řadě jiných variant. Např. původní algoritmus Land and Doig [1960] uvažoval nejprve dvě podúlohy se zafixováním $x_k \coloneqq \lfloor x_k^0 \rfloor$ resp. $x_k \coloneqq \lfloor x_k^0 \rfloor + 1$, a teprve potom v případě potřeby další hodnoty proměnné x_k.

Příklad 3.6. Řešte příklad 2.20 pomocí metody branch & bound.

Řešení: Uvažovaná úloha má tvar

$$\max\; x_1 - x_2 \;\text{ za podm. }\; -\tfrac{1}{3}x_1 + x_2 \le \tfrac{1}{3}, \quad x_1 - \tfrac{1}{3}x_2 \le \tfrac{1}{3}, \quad x \ge 0,\; x \in \mathbb{Z}^2.$$

Podobně jako v příkladu 2.25 vyřešíme nejprve lineární relaxaci ℓ-metodou.

	x_1	x_2	
x_0	-1	1	0
x_1	-1	0	0
x_2	0	-1	0
x_3	$-\frac{1}{3}$	1	$\frac{1}{3}$
x_4	1	$-\frac{1}{3}$	$\frac{1}{3}$
x_5	$\textcircled{1}$	1	1

$\sim$

	x_5	x_2	
x_0	1	2	1
x_1	1	1	1
x_2	0	-1	0
x_3	$\frac{1}{3}$	$\frac{4}{3}$	$\frac{2}{3}$
x_4	$\textcircled{-1}$	$-\frac{4}{3}$	$-\frac{2}{3}$

$\sim$

	x_4	x_2	
x_0	1	$\frac{2}{3}$	$\frac{1}{3}$
x_1	1	$-\frac{1}{3}$	$\frac{1}{3}$
x_2	0	-1	0
x_3	$\frac{1}{3}$	$\frac{8}{9}$	$\frac{4}{9}$
x_4	-1	0	0

Protože řešení $x^0 = (\frac{1}{3}, 0)$ relaxované úlohy nesplňuje celočíselné podmínky, rozdělíme úlohu na dvě podúlohy. Celočíselnost porušuje pouze první proměnná, proto zvolíme $k = 1$. První podúloha vznikne přidáním podmínky $x_k \le \lfloor x_k^0 \rfloor$, jež má nyní podobu $x_1 \le 0$. Pro přidání do ℓ-tabulky je vhodnější vyjádření (3.1), jež má tvar $-x_4 + \frac{1}{3}x_2 \le -\frac{1}{3}$. Přidáme tuto podmínku do ℓ-tabulky a přeoptimalizujeme.

	x_4	x_2	
x_0	1	$\frac{2}{3}$	$\frac{1}{3}$
x_1	1	$-\frac{1}{3}$	$\frac{1}{3}$
x_2	0	-1	0
x_3	$\frac{1}{3}$	$\frac{8}{9}$	$\frac{4}{9}$
x_4	-1	0	0
x_6	$\textcircled{-1}$	$\frac{1}{3}$	$-\frac{1}{3}$

$\sim$

	x_6	x_2	
x_0	1	0	0
x_1	1	0	0
x_2	0	-1	0
x_3	$\frac{1}{3}$	1	$\frac{1}{3}$
x_4	-1	$\frac{1}{3}$	$\frac{1}{3}$
x_6	-1	0	0

Stačila tedy pouze jedna transformace a dostali jsme optimální celočíselné řešení $x^1 = (0, 0)$ v této větvi.

V druhé větvi řešíme podúlohu s přidanou podmínkou $x_k \geq \lfloor x_k^0 \rfloor + 1$, která nabyde tvaru $x_1 \geq 1$. V řeči nebázických proměnných má nerovnice tvar $x_4 - \frac{1}{3}x_2 \leq -\frac{2}{3}$. Přidáme tuto podmínku do ℓ-tabulky a přeoptimalizujeme.

	x_4	x_2	
x_0	1	$\frac{2}{3}$	$\frac{1}{3}$
x_1	1	$-\frac{1}{3}$	$\frac{1}{3}$
x_2	0	-1	0
x_3	$\frac{1}{3}$	$\frac{8}{9}$	$\frac{4}{9}$
x_4	-1	0	0
x_6	1	$-\frac{1}{3}$	$-\frac{2}{3}$

$\sim$

	x_4	x_6	
x_0	3	2	-1
x_1	0	-1	1
x_2	-3	-3	2
x_3	3	$\frac{8}{3}$	$-\frac{4}{3}$
x_4	-1	0	0
x_6	0	-1	0

Protože ve třetím řádku tabulky je napravo záporná hodnota, ale nalevo samé kladné, nelze vybrat žádného pivota, a proto je množina přípustných řešení prázdná. V této větvi výpočetního stromu se tedy optimum nenalézá, a proto je optimálním řešením $x^1 = (0, 0)$. □

3.1 Procházení výpočetního stromu

Binární výpočetní strom lze procházet různými způsoby. Nabízí se základní postupy jako prohledávání do šířky (seznam podúloh je fronta) a do hloubky (seznam podúloh je zásobník). Druhá varianta se zdá perspektivnější, protože rychleji najde přípustné řešení, které v důsledku pomůže ořezat neplodné větve v stromu.

Sofistikovanější možnost je v daném kroku vybrat podúlohu s maximálním horním odhadem na optimální hodnotu (tedy s největší optimální hodnotou relaxace). Jaká datová struktura se hodí pro seznam podúloh? Protože potřebujeme rychle najít maximum a odstranit ho, a umět přidávat další prvky, tak se hodí *halda*, která operace vykonává v logaritmickém čase vzhledem k velikosti seznamu. Teoreticky pak vychází nejlépe *Fibonacciho halda*, kde vkládání trvá amortizovaně konstantní čas.

3.2 Volba proměnné na dělení

Pro efektivitu procházení stromu má velký vliv také volba celočíselné proměnné x_k, podle které vytváříme podúlohy.

Volba má vesměs podobu

$$k := \arg \max_{i \in C : x_i^0 \notin \mathbb{Z}} \min(d_i^-, d_i^+),$$

kde d_i^-, d_i^+ jsou určité míry vhodnosti dané proměnné na dělení.

1. Nejjednodušší volbou je míra neceločíselnosti

   $$d_i^- := \{x_i^0\}, \quad d_i^+ := 1 - \{x_i^0\}.$$

   Volíme tedy proměnnou, která odpovídá nejméně celočíselné složce relaxovaného optima.

2. Další možností je použít pseudoderivace

   $$d_i^- = d_i^+ := |c_i|.$$

   Koeficienty účelové funkce udávají míru změny účelové funkce při změně proměnné, tedy větší vliv na změnu relaxované optimální hodnoty mají proměnné s větší hodnotou $|c_i|$.

3. Kombinací obou předchozích dostáváme

   $$d_i^- := \{x_i^0\}|c_i|, \quad d_i^+ := (1 - \{x_i^0\})|c_i|,$$

   což dává odhad, o kolik se odříznutím sníží optimální hodnota relaxace.

4. O něco dražší, ale přesnější, je pro každého kandidáta použít zaokrouhlení nahoru i dolů a podívat se jak se změní (přesně či aspoň přibližně) hodnota účelové funkce pro provedení jednoho kroku simplexové metody. Pro dané $i \in C$ takové, že $x_i^0 \notin \mathbb{Z}$, i-tý řádek simplexové tabulky říká

$$x_i = R_{i0} - \sum_{j=1}^{n} R_{ij} x_{N_j}.$$

Podmínka $x_i \leq \lfloor x_i^0 \rfloor$ vede na novou nerovnici

$$-\sum_{j=1}^{n} R_{ij} x_{N_j} \leq -\{R_{i0}\},$$

a účelová hodnota po jedné transformaci tabulky klesne aspoň o

$$d_i^- := \min_{j:R_{ij}>0} \frac{R_{0j}}{R_{ij}} \{R_{i0}\}.$$

Analogicky, podmínka $x_i \geq \lfloor x_i^0 \rfloor + 1$ vede na novou nerovnici

$$\sum_{j=1}^{n} R_{ij} x_{N_j} \leq -(1 - \{R_{i0}\}),$$

a účelová hodnota po jedné transformaci tabulky klesne aspoň o

$$d_i^+ := \min_{j:R_{ij}<0} \frac{R_{0j}}{-R_{ij}} (1 - \{R_{i0}\}).$$

3.3 Další varianty

Branch & cut

Branch & cut [Padberg and Rinaldi, 1987] je kombinací obou hlavních metod sečných nadrovin a branch & bound. V zásadě se jedná o branch & bound s tím, že v určitých uzlech výpočetního stromu osekáme relaxovaný vrchol pomocí sečných nadrovin. Většinou se řezy používají u kořene a v blízkých uzlech, abychom měli dobré relaxované řešení hned v počátku.

Nic nám nebrání přidat víc sečných nadrovin najednou. Není moc výhodné přidávat všechny myslitelné řezy, protože řada z nich bude redundantní a akorát navýší paměťovou a tím i časovou náročnost. Achterberg [2009] doporučuje vybrat několik řezů, které jsou na sebe co nejkolmější. Zvolíme-li threshold $\varepsilon > 0$, pak stačí postupovat hladovým algoritmem a přidávat takové sečné nadroviny, jejichž normály (po znormování) mají skalární součin v absolutní hodnotě nanejvýš ε se všemi již vybranými řezy.

Branch & price

Jedná se o kombinaci branch & bound a metody *column generation* pro řešení velkých úloh s obrovským počtem proměnných. Protože bázické optimální řešení lineárního programu má nebázické složky nulové, metoda column generation nejprve vybere jen proměnné do báze a ostatní zafixuje na nulu. Poté iterativně generuje sloupečky tabulky, tj. uvolňuje další proměnné podle *reduced cost*, tj. podle nejpokaženější podmínky na nezápornost kriteriálního řádku.

3.4 Implementační záležitosti

Preprocessing

Uvažme úlohu

$$\max\ c^T x \ \text{ za podm. } \ Ax \leq b,\ l \leq x \leq u.$$

Před vlastním řešením úlohy lze provést následující zjednodušující operace pro každou nerovnici $a^T x \leq b_i$ ze soustavy $Ax \leq b$:

- *Utáhnutí mezí.* Pro každé k

$$a_k > 0 \;\Rightarrow\; x_k \le \frac{1}{a_k}\Big(b_i - \sum_{j\neq k: a_j>0} a_j l_j - \sum_{j\neq k: a_j<0} a_j u_j\Big),$$
$$a_k < 0 \;\Rightarrow\; x_k \ge \frac{1}{a_k}\Big(b_i - \sum_{j\neq k: a_j>0} a_j u_j - \sum_{j\neq k: a_j<0} a_j l_j\Big).$$

- *Test redundance.* Daná nerovnost je redundantní pokud

$$\sum_{j:a_j>0} a_j u_j + \sum_{j:a_j<0} a_j l_j \le b_i.$$

- *Test nepřípustnosti.* Systém je nepřípustný pokud pro nějaké i

$$\sum_{j:a_j>0} a_j l_j + \sum_{j:a_j<0} a_j u_j > b_i.$$

- *Fixace proměnných.* Pro každé k

  (1) pokud $c_k \ge 0$ a $a_{ik} \le 0$ pro všechna i, pak můžeme zafixovat $x_k := u_k$,

  (2) pokud $c_k \le 0$ a $a_{ik} \ge 0$ pro všechna i, pak můžeme zafixovat $x_k := l_k$.

Tyto předzpracující operace fungují dobře i pro lineární programování. V celočíselném programování můžeme navíc při utáhnutí mezí nové meze zaokrouhlit – horní odhad dolů a dolní odhad nahoru. Operace navíc můžeme opakovat, protože výsledek jedné operace ovlivňuje ty ostatní.

Příklad 3.7. Uvažme celočíselný program

$$\begin{aligned}\max\ 2x_1 + x_2 - x_3 \ \text{ za podm. }\ & 5x_1 - 2x_2 + 8x_3 \le 15,\\ & 8x_1 + 3x_2 - 1x_3 \ge 9,\\ & x_1 + x_2 + x_3 \le 6,\\ & 0 \le x_1 \le 3,\\ & 0 \le x_2 \le 1,\\ & 1 \le x_3, \quad x \in \mathbb{Z}^3.\end{aligned}$$

Aplikujte jednotlivé operace preprocessingu: □

Další

Komerční systémy často obsahují

- *Priority.* Možnost zadat priority proměnným, podle kterých se pak řídí větvení ve výpočetním stromu.
- *User cut-offs.* Možnost zadat preference uživatele na *simplex strategies* (zda simplexová metoda či metoda vnitřních bodů), *strong branching* (zda věnovat větší čas dobrému výběru větve, tj. proměnné x_k), atp.
- *GUB (generalized upper bound) branching nebo též SOS1 branching (Beale & Tomlin, 1970).* Řada úloh obsahuje podmínky typu: $\sum_{i=1}^{n} x_i = 1$, $x \in \{0,1\}^n$, to jest, právě jedna z proměnných x_i je jedna, ostatní nulové. Sestavovat podúlohy popsaným způsobem není efektivní, lepší je uvažovat podúlohy:

  (1) $x_i = 0$ pro všechna $i \in C_1$,

  (2) $x_i = 0$ pro všechna $i \in C_2$,

kde $C_1 \cup C_2 = \{1, \ldots, n\}$ je disjunktní rozklad proměnných. Rozklad můžeme provést sekvenčně jako $C_1 = \{1, \ldots, r\}$, $C_2 = \{r+1, \ldots, n\}$, nebo po vhodné permutaci. Každopádně, obě podúlohy mají výrazně menší dimenzi.

Pokud bychom použili formulaci z příkladu 1.35, tak relaxovaná úloha pak má typicky řešení $1 = w_1 = \ldots = w_{k-1} > w_k > w_{k+1} = \ldots = w_n = 0$, tudíž větvení podle neceločíselné proměnné w_k rozdělí úlohu na dvě srovnatelně velké podúlohy, které mají automaticky zafixovaných k resp. $n - k + 1$ binárních proměnných.

Více o tom jak řešit podobné „symetrické“ podmínky viz Margot [2010].

Následující tabulka uvádí klíčové faktory na zrychlení praktických implementací během období kolem roku 2000; od té doby ale heuristiky získaly důležitější roli.

faktor	koeficient zrychlení
řezy	54
preprocessing	11
volba proměnné na dělení	3
heuristiky	1.5

3.5 Aktuální trendy

Aktuální trendy výzkumu (k roku 2015) zahrnují např.

- *Zobecněné větvení* podle obecných nadrovin. Čili místo podle nadroviny $x_i = x_i^0$ rozděluji podúlohu podle nadroviny $a^T x = b$.
- *Výběr LP solveru* na řešení jednotlivých relaxovaných podúloh výrazně ovlivňuje rychlost celého algoritmu.

3.6 Software

Nekomerční solvery:

- *SCIP* (Solving Constraint Integer Programs), `http://scip.zib.de/`
- *GLPK* (GNU Linear Programming Kit), `http://www.gnu.org/software/glpk/`

Komerční solvery (výběr):

- *CPLEX*, `http://www.cplex.com/`
- *Gurobi* (GNU Linear Programming Kit), `http://www.gurobi.com/`

Modelovací systémy zahrnující výběr z výše zmíněných solverů:

- *GAMS* (General Algebraic Modeling System), `http://www.gams.com/`
- *NEOS* (Network Enabled Optimization Server), `http://neos.mcs.anl.gov/neos/` – zdarma na dálku
- *AIMMS* (Advanced Interactive Multidimensional Modeling System), `http://www.aimms.com/`
- *AMPL* (A Mathematical Programming Language), `http://ampl.com/`

Další informace viz např. Atamtürk and Savelsbergh [2005]; Bussieck and Vigerske [2011].

Cvičení

3.1. Dokažte, že při použití metody branch & bound na smíšenou úlohu s jedinou celočíselnou proměnnou navštívíme při procházení větvícího stromu nejvýše 3 uzly.

Kapitola 4

Heuristiky

Heuristiky jsou jednoduché metody, které mají za cíl rychle najít dobré přípustné řešení, což zvyšuje rychlost method typ branch & bound. Pozitivní výsledek nemáme garantovaný, takže heuristiky mohou selhat. Význam dobrých heuristik ale je, že v mnoha případech fungují, a efektivně pomáhají při řešení tím, že dávají globální dolní mez na optimální hodnotu (viz Poznámka 3.2). Pěkný přehled heuristik najdeme v Berthold [2006], a heuristiky používané programem SCIP pak v Berthold [2007].

4.1 Heuristiky pro nalezení přípustného řešení

Relax & fix

Uvažujme úlohu ve tvaru

$$\max\ c^T x + d^T y \ \text{ za podm. } \ Ax + By \leq b,\ x, y \geq 0,\ x \in \mathbb{Z}^n,\ y \in \mathbb{Z}^m.$$

Zde, rozdělení proměnných do dvou skupin vychází z podstaty problému, nebo x jsou „důležitější“ proměnné, které požadujeme, aby byly blíže k optimu.

Heuristika sestává ze dvou kroků. V prvním řešíme relaxovanou úlohu s $x \in \mathbb{Z}^n,\ y \in \mathbb{R}^m$ (tedy relaxuji celočíselnost proměnných y). Nechť dostanu optimální řešení x^*, y^*. V druhém kroku zafixujeme $x \coloneqq x^*$ a dořešíme původní úlohu. Pokud dostaneme přípustné řešení, je dobrým přípustným řešením původní úlohy. V této heuristice řešíme tedy 2 menší celočíselné úlohy.

Cut & fix

Uvažujme úlohu ve tvaru

$$\max\ c^T x + d^T y \ \text{ za podm. } \ Ax + By \leq b,\ x, y \geq 0,\ x \in \mathbb{Z}^n,\ y \in \mathbb{Z}^m.$$

Tato heuristika nejprve k omezením přidá několik sečných nadrovin a spočítáme relaxované řešení $x^* \in \mathbb{R}^n, y^* \in \mathbb{R}^m$. Nyní, pro předem zvolenou malou konstantu $\varepsilon > 0$, přidáme podmínky

$$\lfloor x^* + \varepsilon e \rfloor \leq x \leq \lceil x^* - \varepsilon e \rceil$$

a dořešíme do celočíselného optima. Heuristické řešení získám rychleji, protože přidané podmínky sníží trochu dimenzi úlohy. Pokud totiž x_i^* je dostatečně blízko celému číslu (nanejvýš o ε), pak

$$\lfloor x_i^* + \varepsilon \rfloor = \lceil x_i^* - \varepsilon \rceil,$$

což zafixuje i-tou proměnnou x_i.

Feasibility pump

Základní idea: Najdi optimální řešení relaxované úlohy, zaokrouhli a najdi nejbližší přípustný bod relaxované úlohy v součtové normě. Postup iteruj. Detaily viz Achterberg and Berthold [2007].

4.2 Heuristiky pro nalezení dobrého řešení

Nyní se zaměříme na úlohy, kde je snadné najít přípustná řešení, ale je velmi těžké najít optimum. Následující heuristiky se proto snaží najít co nejlepší přípustné řešení (bez garance optimality) a jsou adaptací z heuristik pro úlohy spojité optimalizace s velkou dimenzí.

Lokální prohledávání

Tato heuristika funguje tak, že dané přípustné řešení se snaží lokálně vylepšit. Prohledá sousední řešení a pokud najde lepší, změní aktuální přípustné řešení na toto nalezené a postup opakuje.

Konkrétní postup se dost odvíjí od konkrétní úlohy. V problému obchodního cestujícího může jít o tzv. dvoj-výměny, kdy danou cestu na dvou místech přestřihnu a spojím opačné konce (viz sekce 9.4). Heuristiky pro úlohu Facility location problem jsou v sekci 10.3.

Následující heuristiky se snaží vyvarovat toho, abychom uvízli v lokálním maximu.

Simulované žíhání

Metoda, kdy procházím mezi přípustnými řešeními. Na začátku je symbolická teplota T vysoká a dovoluje nám přejít i k horšímu řešení. Postupem času teplota klesá a nedovolí nám příliš si pohoršit účelovou hodnotu. Až nakonec smíme přecházet jen k lepším řešením.

Formálněji, je-li Δ rozdíl účelových hodnot aktuálního řešení a náhodně vybraného sousedního, tak k sousednímu přejdu vždy pokud $\Delta \geq 0$ (tj., sousední řešení není horší) a s pravděpodobností $e^{-\Delta/T}$, kde T je čas, pokud $\Delta < 0$ (tj., sousední řešení je horší).

Tabu search

Opět procházíme mezi přípustnými řešeními. Pokud se nemohu přemístit k lepšímu řešení, mohu přejít i k horšímu. K zabránění vrácení se zpět k již nalezeným lokálně optimálním řešením slouží tzv. *tabu list*, což je seznam několik předchozích lokálně optimálních řešení, které již nesmím navštívit.

Genetické algoritmy

Jejich princip spočívá v tom, že máme seznam několika přípustných řešení, nazývaný *výběr z populace*. Každý jedinec je nějak kódován. Tyto jedince pak měním podle pravidel

- *mutace* – změním trochu kód jedince,
- *křížení* – zkombinuji kódy dvou jedinců,
- *přírodní výběr* – vybírám co nejlepší jedince.

Implementace těchto pravidel závisí na konkrétním typu úlohy.

Například pro problém obchodního cestujícího, viz sekce 9.4.

Kapitola 5

Dekompozice

Uvedeme si dva typy dekompozic. První, Bendersova, je vertikální, to jest, rozděluje proměnné do dvou částí určitým způsobem redukuje problém na několik menších. Druhá, Lagrangeova, je naopak horizontální a rozděluje omezení do dvou skupin.

5.1 Bendersova dekompozice

Bendersova dekompozice Benders [1962, 2005] funguje v zásadě i pro lineární či speciální nelineární program, ale pro naše účely uvažujme smíšenou úlohu celočíselného programování

$$\max\ c^T x + d^T y \ \text{ za podm. } \ Ax + By \le b,\ x \ge 0,\ y \in Y, \tag{5.1}$$

kde množina Y zahrnuje lineární a celočíselné podmínky jako např. $y \ge 0$, $y \in \mathbb{Z}^n$ atp. Budeme předpokládat $Y \neq \emptyset$. Potom můžeme úlohu přepsat jako

$$\max_{y \in Y} \left\{ d^T y + \max\{c^T x;\ Ax \le b - By,\ x \ge 0\} \right\},$$

nebo pomocí duálního lineárního programu jako

$$\max_{y \in Y} \left\{ d^T y + \min\{(b - By)^T u;\ A^T u \ge c,\ u \ge 0\} \right\}.$$

Speciálně, primární a duální lineární program označíme jako

$$\max\{c^T x;\ Ax \le b - By,\ x \ge 0\}, \tag{$P(y)$}$$
$$\min\{(b - By)^T u;\ A^T u \ge c,\ u \ge 0\}. \tag{$D(y)$}$$

Abychom mohli použít dualitu, budeme ještě předpokládat, že množina přípustných řešení duální úlohy

$$U \coloneqq \{u;\ A^T u \ge c,\ u \ge 0\} \neq \emptyset.$$

Neprázdnost U snadno ověříme zvlášť, a pokud by neplatila, tak původní úloha je neomezená či prázdná. Pro praktické úlohy je to vesměs splněno, jinak lze teoreticky použít nástroje popsané např. v Poznámce 1.24.

Buďte $v_1, \dots, v_V$ vrcholy a $h_1, \dots, h_H$ směry neomezených hran konvexního polyedru U. Pro pevné y, pokud v úloze $D(y)$ nastane $(b - By)^T h_j < 0$ pro jisté j, pak $D(y)$ je neomezená a tudíž $P(y)$ nepřípustná. Proto se stačí omezit na situace kdy $(b - By)^T h_j \ge 0$ pro všechna $j = 1, \dots, H$. Nyní můžeme úlohu (5.1) ekvivalentně vyjádřit ve tvaru

$$\max \left(d^T y + \min_{i=1,\dots,V} (b - By)^T v_i \right) \ \text{ za podm. } \ (b - By)^T h_j \ge 0\ \forall j, \ \ y \in Y,$$

neboli

$$\max\ z \ \text{ za podm. } \ z \le d^T y + (b - By)^T v_i\ \forall i, \ \ (b - By)^T h_j \ge 0\ \forall j, \ \ y \in Y. \tag{5.2}$$

Tímto jsme zmenšili dimenzi původní úlohu, protože namísto spojitých proměnných x zde máme již jen jednu spojitou proměnnou z. Nicméně, celočíselné proměnné zůstávají stejné.

Abychom nemuseli enumerovat všechny vrcholy a neomezené hrany U, tak je budeme generovat postupně dle potřeby podle následujícího iterativního algoritmu. Pro indexové množiny I, J si označme

$$\max\ z \quad \text{za podm. } z \le d^T y + (b - By)^T v_i\ \forall i \in I,\ \ (b - By)^T h_j \ge 0\ \forall j \in J,\ \ y \in Y. \qquad P_{I,J}$$

Algoritmus 5.1 (Bendersova dekompozice)**.** Na začátku polož $I := \emptyset$, $J := \emptyset$. V k-té iteraci postupujeme

1: Řeš úlohu $P_{I,J}$. Má-li úloha optimum (y^*, z^*), jdi na krok 2.

Je-li úloha neomezená, tak (y^*, z^*) zvol jako vrchol z něhož vychází neomezená hrana a jdi na krok 2.

Je-li nepřípustná, je nepřípustná i (5.1). Konec.

2: Řeš úlohu $D(y)$ s $y := y^*$.

Nepřípustnost nenastane díky $U \neq \emptyset$.

Je-li neomezená dle hrany h_{j^*} vycházející z vrcholu v_{i^*}, tak přiřaď $I := I \cup \{i^*\}$, $J := J \cup \{j^*\}$ a jdi na krok 1.

Má-li optimální řešení ve vrcholu v_{i^*}, tak rozlišme dva případy:

(a) Je-li $z^* \le d^T y^* + (b - By^*)^T v_{i^*}$, pak platí $z^* \le d^T y^* + (b - By^*)^T v_i$ pro všechna $i = 1, \ldots, V$ a proto je (y^*, z^*) optimum úlohy (5.2). Tudíž je z^* optimální hodnota i původní úlohy (5.1), y^* je optimální řešení a druhou část optima x^* dopočítáme z podmínek duality.

(b) Jinak, přiřaď $I := I \cup \{i^*\}$, a jdi na krok 1.

Konečnost algoritmu je zaručena, počet iterací je maximálně $V + H$.

Navíc, algoritmus lze kdykoli ukončit, a máme dolní a horní odhady na optimální hodnoty. Jako horní odhad optimální hodnoty slouží z^* (postupem iterací se zmenšuje). Naopak, pokud úloha $D(y)$ má optimum v kroku 2 Algoritmu 5.1, tak $z^\circ := d^T y^* + (b - By^*)^T v_{i^*}$ je dolní mezí optimální hodnoty (neboť (y^*, z°) je přípustné řešení).

Bendersova dekompozice se nevyplatí pro každou úlohu, ale pro některé typy úloh je vhodná. Ukážeme si její použití v úloze UFLP (sekce 10.4).

Poznámka 5.2. Bendersovou dekompozicí dostaneme úlohu se stejným počtem celočíselných proměnných, ale s pouze jedinou spojitou proměnnou z. Protože čistě celočíselné úlohy se řeší snadněji, může být výhodné proměnnou z uvažovat jako celočíselnou a pak binárním půlením aproximovat spojitou hodnotu.

Příklad 5.3. Bendersovou dekompozicí vyřešte úlohu (viz obrázek 5.1)

$$\begin{aligned} \max\ -5x + 3y\ \ &\text{za podm.} \\ -2x + y &\le 0, \\ -x + 3y &\le 13, \\ y &\le 10, \\ x, y \ge 0,\ \ y &\in \mathbb{Z}. \end{aligned}$$

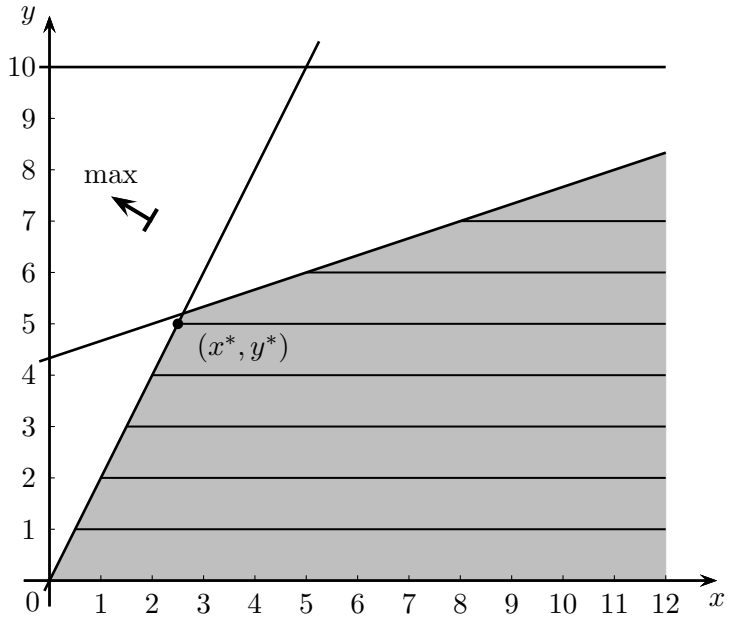
Řešení: V této úloze je

$$c = (-5),\ \ d = (3),\ \ A = \begin{pmatrix} -2 \\ -1 \end{pmatrix},\ \ B = \begin{pmatrix} 1 \\ 3 \end{pmatrix},\ \ b = \begin{pmatrix} 0 \\ 13 \end{pmatrix},$$

a množina U representuje trojúhelník s vrcholy $(0,0)^T$, $(0,5)^T$ a $(2.5,0)^T$:

$$U = \{u \in \mathbb{R}^2;\ -2u_1 - u_2 \ge -5,\ u \ge 0\}.$$

Na začátku nastavíme $I := \emptyset$, $J := \emptyset$. Jednotlivé iterace jsou:



Obrázek 5.1: Množina přípustných řešení, účelová funkce a optimální řešení (x^*, y^*) příkladu 5.3.

1. Nejprve řešíme úlohu
$$\max\ z\ \text{ za podm. }\ y \in Y = \{y \le 10,\ y \ge 0,\ y \in \mathbb{Z}\}.$$
Úloha je neomezená, volíme $y^* = 0^T$, $z^* = \infty$. Nyní řešíme úlohu
$$\min_{u \in U}\ (b - By)^T u = \min_{u \in U}\ 0u_1 + 13u_2.$$
Úloha má optimum v bodě $v_1 = (0,0)^T$. Aktualizujeme $I = \{1\}$.
2. Řešíme úlohu
$$\max\ z\ \text{ za podm. }\ y \in Y,\ z \le d^T y + (b - By)^T v_1,$$
čili
$$\max\ z\ \text{ za podm. }\ y \in Y,\ z \le 3y.$$
Úloha má optimum $y^* = 10$, $z^* = 30$. Nyní řešíme úlohu
$$\min_{u \in U}\ -10u_1 - 17u_2.$$
Úloha má optimum ve vrcholu $v_2 = (0,5)^T$. Aktualizujeme $I = \{1,2\}$.
3. Řešíme úlohu
$$\max\ z\ \text{ za podm. }\ y \in Y,\ z \le 3y,\ z \le 65 - 12y.$$
Úloha má optimum $y^* = 4$, $z^* = 12$. Úloha
$$\min_{u \in U}\ -4u_1 + u_2$$
má optimum ve vrcholu $v_3 = (2.5,0)^T$. Aktualizujeme $I = \{1,2,3\}$.
4. Řešíme úlohu
$$\max\ z\ \text{ za podm. }\ y \in Y,\ z \le 3y,\ z \le 65 - 12y,\ z \le 0.5y.$$
Úloha má optimum $y^* = 5$, $z^* = 2.5$. Úloha
$$\min_{u \in U}\ -5u_1 - 2u_2$$
má optimum ve vrcholu $v_3 = (2.5,0)^T$ a platí
$$2.5 = z^* \le d^T y^* + (b - By^*)^T v_3 = 2.5.$$
Tudíž máme optimum původní úlohy: Optimální hodnota je 2.5, optimální řešení je $y^* = 5$ a $x^* = 2.5$ (to zjistíme vyřešením lineárního programu po dosazení y^* do původní úlohy). □

Cvičení

5.1. Bendersovou dekompozicí vyřešte úlohu

$$\begin{aligned} \max\ -2x_1 - 2x_2 - 3y_1 - 4y_2; \\ x_1 - x_2 - 3y_1 + 1y_2 \le -6, \\ -x_1 + 3x_2 - 2y_1 - 2y_2 \le -5, \\ y_1 \le 2, \\ x, y \ge 0, \quad y_1, y_2 \in \mathbb{Z}. \end{aligned}$$

5.2 Lagrangeova relaxace a dekompozice

Lagrangeova relaxace

Snaha Lagrangeova relaxace (z roku 1963) je získat lepší relaxaci a tím i horní odhad na optimální hodnotu než obyčejnou celočíselnou relaxací.

Uvažujme celočíselný program ve tvaru

$$\max\ c^T x \ \text{ za podm. } \ Ax \le b,\ Dx \le d,\ x \ge 0,\ x \in \mathbb{Z}^n, \tag{P}$$

kde $A \in \mathbb{R}^{m\times n}$, $D \in \mathbb{R}^{\ell\times n}$, $b \in \mathbb{R}^m$, $d \in \mathbb{R}^\ell$. Připomeňme, že standardní celočíselná relaxace má tvar

$$\max\ c^T x \ \text{ za podm. } \ Ax \le b,\ Dx \le d,\ x \ge 0,\ x \in \mathbb{R}^n. \tag{CR}$$

Pro účely Lagrangeovy relaxace předpokládejme, že podmínky $Ax \le b$ jsou „nehezké“, a ty ostatní jsou „hezké“ v tom smyslu, že na množině

$$Q \coloneqq \{x \in \mathbb{R}^n;\ Dx \le d,\ x \ge 0,\ x \in \mathbb{Z}^n\}$$

se snadno optimalizuje (například, že matice D je totálně unimodulární). Rozdělení podmínek pak často vychází z podstaty daného problému.

Definujme úlohu Lagrangeovsky relaxující podmínku $Ax \le b$ jako

$$\max\ c^T x + u^T(b - Ax) \ \text{ za podm. } \ x \in Q. \tag{P_u}$$

Tvrzení 5.4. *Platí $(P) \le (P_u)$ pro každé $u \ge 0$.*

Důkaz. Buď x přípustné řešení úlohy (P) a $u \ge 0$. Pak x je přípustným řešením úlohy (P_u) a navíc $c^T x \le c^T x + u^T(b - Ax)$. □

Tedy úloha (P) je relaxací úlohy (P_u). Podobně relaxujeme i úlohy v jiném tvaru, např. pro rovnice není nutná nezápornost u.

Protože (P) relaxuje (P_u) pro všechna $u \ge 0$, je přirozené vzít to nejlepší možné $u \ge 0$, což nás vede na *Lagrangeovu relaxaci*

$$\min\ (P_u) \ \text{ za podm. } \ u \ge 0. \tag{LR}$$

Je to vlastně Lagrangeova duální úloha z teorie nelineární optimalizace.

Jaká je síla Lagrangeovy relaxace? Vždy je aspoň tak dobrá jako celočíselná relaxace:

Věta 5.5. *Platí $(LR) = \max\{c^T x;\ Ax \le b,\ x \in \operatorname{conv}(Q)\}$.*

Důkaz. Bez újmy na obecnost (srov. poznámku 1.24) nechť Q je omezený a skládá se z bodů $x_1, \dots, x_q$. Pak Lagrangeova relaxace (P_u) má tvar

$$\begin{aligned} \min_{u\ge 0}\ (P_u) &= \min_{u\ge 0}\ \max_{i=1,\dots,q}\ c^T x_i + u^T(b - Ax_i) \\ &= \min\ z \quad \text{za podm. } \ z \ge c^T x_i + u^T(b - Ax_i)\ \forall i,\ u \ge 0,\ z \in \mathbb{R}. \end{aligned}$$

Toto je přípustná úloha lineárního programování, tudíž ji můžeme nahradit duální úlohou

$$\max \ \textstyle\sum_{i=1}^{q}(c^T x_i)y_i \ \text{ za podm. } \ \sum_{i=1}^{q}(b - Ax_i)y_i \geq 0,\ e^T y = 1,\ y \geq 0.$$

Výraz $\sum_{i=1}^{q} y_i x_i$ vyjadřuje konvexní kombinaci bodů z Q. Takže substitucí $x \coloneqq \sum_{i=1}^{q} y_i x_i$ dostaneme úlohu

$$\max \ c^T x \ \text{ za podm. } \ Ax \leq b,\ x \in \operatorname{conv}(Q). \qquad \square$$

Důsledek 5.6. *Platí* $(P) \leq$ *(LR)* $\leq$ *(CR).*

V řetízku nerovností mohou nastat všechny možné kombinace kdy se nějaká nerovnost nabyde jako rovnost či jako ostrá nerovnost. Ukážeme postačující podmínky pro tyto situace.

Důsledek 5.7. *Je-li* $\operatorname{conv}(Q) = \{x \in \mathbb{R}^n;\ Dx \leq d,\ x \geq 0\}$*, pak (LR) = (CR).*

Lagrangeova relaxace je tudíž stejná jako klasická celočíselná relaxace pokud je polyedr $\{x \in \mathbb{R}^n;\ Dx \leq d,\ x \geq 0\}$ celočíselný. To nastane například pokud matice D je totálně unimodulární. Tudíž aby Lagrangeova relaxace byla silnější než celočíselná relaxace, nesmí se na množině Q optimalizovat až příliš snadno.

Naopak, následující postačující podmínka nám zaručuje *(LR)* $= (P)$.

Tvrzení 5.8. *Buď* $u \geq 0$*, buď* x^* *optimální řešení* (P_u) *takové, že* $Ax^* \leq b$ *a navíc* $(Ax)_i = b_i$ *pro všechna* $i : u_i > 0$*. Pak* x^* *je optimálním řešením* (P)*.*

Důkaz. Podle předpokladu je x^* přípustné řešení úlohy (P). Podle tvrzení 5.4 je $(P) \leq c^T x^* + u^T(b - Ax^*) = c^T x^*$. Tudíž x^* je optimálním řešením (P). $\square$

Subgradientní algoritmus

Jak řešit úlohu *(LR)*? Ukážeme si efektivnější metodu než prostou enumeraci z důkazu věty 5.5.

Z důkazu věty 5.5 můžeme úlohu *(LR)* vyjádřit jako $\min_{u \geq 0} f(u)$, kde

$$f(u) \coloneqq \max_{i=1,\dots,q} (b - Ax_i)^T u + c^T x_i, \tag{5.3}$$

a $x_1, \dots, x_q$ jsou prvky množiny Q. Tato funkce je konvexní a po částech lineární. Tudíž nemůžeme použít metody konvexní (hladké) optimalizace a musíme na to jít jinak. Konkrétně, využijeme subgradientů.

Definice 5.9. *Subgradientem* konvexní funkce $f\colon \mathbb{R}^n \mapsto \mathbb{R}$ v bodě u^* je libovolný vektor $d \in \mathbb{R}^n$ takový, že $f(u) \geq f(u^*) + d^T(u - u^*)$ pro všechna $u \in \mathbb{R}^n$.

Geometricky, $f(u)$ musí ležet nad nadrovinou $f(u^*) + d^T(u - u^*)$, která se f dotýká v bodě u^*. Zřejmě, je-li f diferencovatelná, tak gradientem je (a musí být) gradient $d = \nabla f(u^*) = \left(\frac{\partial f}{\partial u_1}(u^*), \dots, \frac{\partial f}{\partial u_n}(u^*)\right)^T$.

Sugradient naší funkce (5.3) spočítáme podle následujícího pozorování.

Tvrzení 5.10. *Buď* x_{u^*} *optimální řešení* (P_u) *s* $u \coloneqq u^*$*, to jest*

$$f(u^*) \coloneqq (b - Ax_{u^*})^T u^* + c^T x_{u^*}.$$

Pak funkce $f(u)$ *má v bodě* u^* *subgradient* $d \coloneqq b - Ax_{u^*}$*.*

Důkaz. Buď $u \geq 0$. Máme ukázat $f(u) \geq f(u^*) + d^T(u - u^*)$, neboli

$$\max_{i=1,\dots,q} (b - Ax_i)^T u + c^T x_i \geq (b - Ax_{u^*})^T u^* + c^T x_{u^*} + (b - Ax_{u^*})^T (u - u^*).$$

To se zjednodušením pravé strany upraví na

$$\max_{i=1,\dots,q} (b - Ax_i)^T u + c^T x_i \geq (b - Ax_{u^*})^T u + c^T x_{u^*},$$

což zjevně platí. $\square$

Základní myšlenka subgradientního algoritmu, podobně jako u gradientních metod, je iterativně se posouvat v množině přípustných řešení ve směru klesání $-d$.

Algoritmus 5.11 (Subgradientní algoritmus)**.** Zvol počáteční u^1 a nastav $k = 1$. V k-té iteraci provedeme:

1: Řeš úlohu (P_u) s $u \coloneqq u^k$, a nechť má optimum x_{u^k}.
2: $u^{k+1} \coloneqq \max(u^k - \alpha_k(b - Ax_{u^k}), 0)$,
3: $k \coloneqq k + 1$.

Druhý krok zaručuje $u^{k+1} \geq 0$.
Parametr $\alpha_k > 0$ určuje délku kroku ve směru klesajícího subgradientu. Možné volby:

- Volíme α_k tak, aby $\alpha_k \to_{k\to\infty} 0$ a $\sum_{k=1}^{\ell} \alpha_k \to_{\ell\to\infty} \infty$. Toto zaručuje konvergenci. Typickou volbou je $\alpha_k = 1/k$, prakticky je to ale pomalé.
- Hodnota α_k je po dobu p iterací konstantní, a pak se po každých p iteracích pronásobí parametrem $\rho \in (0, 1)$. Konvergence je zaručena pro α_1 a ρ dostatečně velké.
- Další volby viz Wolsey [1998].

Ukončovací podmínkou může být např. $d = 0$, překročení limitu počtu iterací, nebo pro celočíselná data podmínka $f(u^k) - z \leq 1$, kde z je účelová hodnota nejlepšího dosud nalezeného přípustného řešení (tedy je to optimum). Poznamenejme, že algoritmus můžeme kdykoliv ukončit a podle věty 5.4 dává (P_u) horní odhad na optimální hodnotu; tedy v principu není nutné řešit *(LR)* do optima za každou cenu.

Lagrangeova dekompozice

Přepišme (P) do tvaru

$$\max\ c^T x \ \text{ za podm. } \ Ax \leq b,\ Dy \leq d,\ x = y,\ x, y \in \mathbb{Z}^n_+.$$

Lagrangeovou relaxací podmínky $x = y$ dostaneme

$$\begin{aligned} &\min_{u\in\mathbb{R}^n} \max\{(c-u)^T x + u^T y;\ Ax \leq b,\ Cy \leq d,\ x, y \in \mathbb{Z}^n_+\} \\ &= \min_{v,w:c=v+w} \left(\max\{v^T x;\ Ax \leq b,\ x \in \mathbb{Z}^n_+\} + \max\{w^T y;\ Dx \leq d,\ x \in \mathbb{Z}^n_+\}\right). \qquad \textit{(LR')} \end{aligned}$$

Takto dostaneme ještě lepší horní mez než přímou Lagrangeovou relaxací.

Tvrzení 5.12. *Platí (LR')* $= \max\{c^T x;\ x \in \operatorname{conv}\{x \in \mathbb{Z}^n_+;\ Ax \leq b\} \cap \operatorname{conv}\{x \in \mathbb{Z}^n_+;\ Dx \leq D\}\}$.

Lagrangeova relaxace pro speciální úlohy

Lagrangeova relaxace je výhodná pro řadu speciálních úloh. V Sekci 9.3 ukážeme použití pro problém obchodního cestujícího a v Sekci 10.5 pro úlohu UFLP.

Cvičení

5.2. Diskutujte obrácenou implikaci v Důsledku 5.7. To jest, pokud *(LR)* = *(CR)*, platí $\operatorname{conv}(Q) = \{x \in \mathbb{R}^n;\ Dx \leq d,\ x \geq 0\}$?

Kapitola 6

Column generation

Tato technika byla vyvinuta pro celočíselné a spojité lineární programy s velkým počtem proměnných. Uvažujme nejprve lineární program

$$\min\ c^T x\ \text{ za podm. }\ Ax = b,\ x \geq 0, \tag{6.1}$$

kde počet proměnných n je velké. Buď $I \subset \{1, \ldots, n\}$. Z matice A vybereme sloupce indexované množinou I, a označíme ji A_I. Jinými slovy, zafixujeme proměnné $x_i = 0$, $i \notin I$. Dostaneme menší podúlohu

$$\min\ c_I^T x_I\ \text{ za podm. }\ A_I x_I = b,\ x_I \geq 0. \tag{6.2}$$

Buď x_I^* optimum této podúlohy. Z podmínek duality snadno získáme i optimum y^* duální úlohy

$$\max\ b^T y\ \text{ za podm. }\ A_I^T y \leq c_I.$$

Doplněním nul vektoru x_I^* dostaneme přípusté řešení x^* původní úlohy (6.1). Potom x^* je optimem, pokud y^* splňuje všechny podmínky duální úlohy k (6.1)

$$\max\ b^T y\ \text{ za podm. }\ A^T y \leq c.$$

Pokud y^* nesplňuje všechny podmínky, tak vyber tu nejvíce pokaženou a přidej její index do seznamu I a přeoptimalizuj podúlohu. Pokud řešíme podúlohu (6.2) primární simplexovou metodou, tak úlohu s přidaným sloupcem snadno přeoptimalizujeme rozšířením simplexové tabulky o jeden sloupec a pivotizací v tomto sloupci.

Pokud v úloze (6.1) máme navíc podmínky na celočíselnost $x \in \mathbb{Z}^n$, tak metodu můžeme zkombinovat s branch & bound.

Příklad 6.1. Uvažujme úlohu ve tvaru

$$\min\ c_1^T x_1 + c_2^T x_2\ \text{ za podm. }\ A_1 x_1 + A_2 x_2 = b,\ x_1, x_2 \geq 0,\ x_i \in X_i,\ i \in \{1, 2\},$$

kde $X_i = \{x_i \in \mathbb{Z}^{n_i};\ C_i x_i \leq d_i\}$, $i \in \{1, 2\}$. Buďte $v_1^i, \ldots, v_{m_i}^i$ všechna přípustná řešení X_i. Pak použijeme substituci

$$x_i = \sum_{k=1}^{m_i} \lambda_k^i v_k^i,\quad \lambda_k^i \geq 0,\quad \sum_{k=1}^{m_i} \lambda_k^i = 1.$$

Tím dostaneme lineární program s potenciálně velmi mnoha proměnnými v_k^i.

Jaká je síla takové transformace? Přepisem vlastně řešíme úlohu

$$\min\ c_1^T x_1 + c_2^T x_2\ \text{ za podm. }\ A_1 x_1 + A_2 x_2 = b,\ x_1, x_2 \geq 0,\ x_i \in \operatorname{conv}(X_i).$$

Tedy určitě je relaxace aspoň tak silná jako standardní celočíselná relaxace, na druhou stranu nemáme garanci, že řešení bude celočíselné. □

Příklad 6.2 (Boland and Surendonk [2001])**.** Uvažujme úlohu plánování rozvozu zboží zákazníkům pro období 12 následujících měsíců. Standardní model by byl

$$\min \sum_{i,s,t} c_{i,s} x_{i,s,t},$$

kde $c_{i,s}$ je cena za přepravu zboží zákazníkovi i lodí s. Proměnná $x_{i,s,t} \in \{0,1\}$ říká, jestli loď s v čase t obslouží zákazníka i. Podmínky jsou:

$$\sum_{s} z_s x_{i,s,t} \geq d_{i,t}, \quad \forall i,t,$$

kde z_s je kapacita lodě s, a $d_{i,t}$ je požadavek zákazníka i v měsíci t. Dále,

$$l_s \leq \sum_{i,t} x_{i,s,t} \leq u_s, \quad \forall s,$$

kde l_s a u_s je minimální a maximální počet použití lodě s za celou dobu rozvrhu. Loď s může být využita maximálně jednou během sekvence m_s měsíců, což vede na podmínky

$$\sum_{i} (x_{i,s,t} + x_{i,s,t+1} + \ldots + x_{i,s,t+m_s-1}) \leq 1, \quad \forall t,s.$$

Konkrétní problém s 16 zákazníky a 30 loděmi nebyl v Cplexu vůbec dořešen do optima a nejlepší nalezené řešení bylo 16% od optima. Celočíselná relaxace je totiž velmi slabá.

Zkusme problém přeformulovat. Zavedeme proměnné $y_{i,k,t} \in \{0,1\}$, které říkají zda požadavek zákazníka i v měsíci t bude uspokojen kombinací lodí k. Úloha má nyní tvar

$$\min \sum_{i,k,t} \left(\sum_{s \in k} c_{i,s} \right) y_{i,k,t},$$

za podmínek

$$l_s \leq \sum_{i,\, k \ni s,\, t} y_{i,s,t} \leq u_s, \quad \forall s,$$

$$\sum_{i,\, k \ni s} (y_{i,k,t} + y_{i,k,t+1} + \ldots + y_{i,k,t+m_s-1}) \leq 1, \quad \forall t,s.$$

Počet proměnných je nyní velmi vysoký, a proto je vhodné použít metodu Column generation. Vybírání a přidávání sloupce zde odpovídá (pro daného zákazníka i a měsíc t) nalezení kombinace lodí k, která nejvíce zlepší účelovou hodnotu. Pro výše zmíněný konkrétní problém byla úloha vyřešena za 2 minuty s tím, že bylo vygenerováno 1700 sloupců (tj., kombinací lodí). Navíc řešení bylo rovnou celočíselné. □

Část II

Speciální případy

Kapitola 7

Problém batohu

Uvažujme problém batohu ve tvaru

$$\max\ c^T x\ \text{ za podm. }\ a^T x \le b,\ x \in \{0,1\}^n, \tag{7.1}$$

kde $a, c \in \mathbb{Z}^n$ a $b \in \mathbb{Z}$. Interpretace může být taková, že chceme maximálně naplnit pomyslný batoh abychom nepřekročili jeho nosnost, přičemž každý předmět má svou hmotnost a užitek.

Předpoklad. Budeme předpokládat, že $a > 0$, $c > 0$. Tento předpoklad je bez újmy na obecnost.

Důkaz. „*Předpoklad* $a > 0$.“ Je-li $a_i = 0$, pak zafixujeme $x_i := 2\,\mathrm{sgn}(c_i) - 1$. Je-li $a_i < 0$, pak zavedeme substituci $x'_i := 1 - x_i \in \{0,1\}$. Koeficienty se pak změní takto: $a'_i := -a_i$, pravá strana nerovnosti $b' := b - a_i$ a koeficient v účelové funkci na $c'_i := -c_i$. Celková cena se zvětší o c_i.

„*Předpoklad* $c > 0$.“ Je-li $c_i \le 0$, pak zafixujeme $x_i := 0$. □

Předpoklad 2. Budeme předpokládat, že $\max_{i=1,\dots,n} a_i \le b$. Tento předpoklad je bez újmy na obecnost.

Důkaz. Kdyby $a_i > b$ pro nějaké i, tak mohu zafixovat $x_i := 0$. □

Další typy. Kromě (7.1) lze uvažovat ještě řadu dalších variant, např.

$$\max\ c^T x\ \text{ za podm. }\ a^T x \le b,\ x \ge 0,\ x \in \mathbb{Z}^n, \tag{7.2}$$

$$\max\ c^T x\ \text{ za podm. }\ a^T x = b,\ x \in \{0,1\}^n. \tag{7.3}$$

Některé uvedené výsledky fungují analogicky i pro tyto varianty, jiné nikoli. Na některé případné odlišnosti upozorňujeme v textu, jiné necháváme čtenáři na rozmyšlenou.

Příklad 7.1 (Investování kapitálu)**.** Uvažujme jednoduchý model investování kapitálu b jednotek peněz mezi m projektů, přičemž i-tý projekt vyžaduje počáteční investici a_i peněz a čistý výnos je c_i peněz. Úloha najít nejvýnosnější portfolio pak vede přímo na tvar (7.1). □

Příklad 7.2 (Problém dělení materiálu)**.** Představme si materiál vyráběný ve velkých kusech (role textilu, ocelové trubky aj.) délky b, ale využitkujeme pouze díly o délkách a_i, $i = 1, \dots, n$. Chceme-li maximálně využít materiál a mít tak minimální odpad, řešíme úlohu (7.2)

$$\max\ a^T x\ \text{ za podm. }\ a^T x \le b,\ x \ge 0,\ x \in \mathbb{Z}^n.$$ □

Preprocessing I.

Je-li $\sum_i a_i \le b$, pak optimum je $x = 1$. Je-li $a_i > b$ pro nějaké i, pak můžeme zafixovat $x_i := 0$.

7.1 Složitost

Věta 7.3 (Korte & Schrader, 1981)**.** *Problém batohu je NP-úplný (všechny varianty).* □

Přestože je problém batohu je NP-úplný, není silně NP-úplný. To protože existuje pseudopolynomiální algoritmus, který běží polynomiálně při unárním kódování vstupu. Jinými slovy, je polynomiální vzhledem k velikosti vstupu (při standardním kódování) a největšímu číslu na vstupu (což v našem případě můžeme brát b). Důsledkem je, že pro malé b poběží pseudopolynomiální algoritmus rychle.

Pseudopolynomiálních algoritmů pro problém batohu existuje povícero, ukážeme si jeden z nich využívajícího principu dynamického programování.

Algoritmus 7.4 (Pseudopolynomiální algoritmus I.)**.** Mějme pole S délky $b+1$, indexováno od 0, udávající v $S[\alpha]$ nejlepší cenu $\sum_{i=1}^{k} c_i x_i$ pro daný součet hmotností $\alpha \coloneqq \sum_{i=1}^{k} a_i x_i$. Na počátku nastav $S[0] \coloneqq 0$, ostatní jsou -1.

Algoritmus sestává ze základního cyklu o n iteracích. V k-té iteraci přidáváme k-tou proměnnou a prozkoumáme jestli můžeme dosáhnout dalšího součtu hmotností a/nebo lepší ceny. To provedeme tak, že pro každé $i = b - a_k, \ldots, 1, 0$ takové, že $S[i] \geq 0$, zjistíme, co způsobí přidání dalšího předmětu. Jestliže $S[i] + c_k > S[i + a_k]$, dosadíme $S[i + a_k] \coloneqq S[i] + c_k$.

Nakonec z pole S vybereme největší hodnotu, která udává optimální hodnotu úlohy. Pokud si u jednotlivých nezáporných složek pole S pamatujeme i index k, můžeme jednoduše zrekonstruovat jak jsme největší hodnotu dostali a jaké (resp. jaká všechna) optimální řešení jí odpovídá.

Příklad 7.5. Adaptujete algoritmus i na ostatní varianty problému batohu (7.2) a (7.3).

Řešení: Pro (7.2) bude jediná změna v testu přidání k-tého předmětu, který mohu přidávat vícekrát. Čili nový test bude probíhat pro $i = 0, 1, \ldots, b - a_k$, protože to umožní opakovaně přidávat jeden předmět.

Pro (7.3) bude naopak jediná změna v nalezení optimální hodnoty, která se nachází v $S[b]$. Je-li $S[b] = -1$, pak úloha nemá přípustné řešení. □

Příklad 7.6. Pomocí Algoritmu 7.4 vyřešte úlohu

$$\max\ x_1 + 5x_2 + 3x_3 + x_4 + 2x_5 \ \text{ za podm. } \ 3x_1 + 4x_2 + 3x_3 + 2x_4 + x_5 \leq 7,\ x \in \{0,1\}^5.$$

Řešení: V tabulce dole je stav pole S na začátku a v průběhu jednotlivých iterací:

indexy pole S	0	1	2	3	4	5	6	7
poč. stav	0	−1	−1	−1	−1	−1	−1	−1
přidání x_1	0	−1	−1	1	−1	−1	−1	−1
přidání x_2	0	−1	−1	1	5	−1	−1	6
přidání x_3	0	−1	−1	3	5	−1	4	8
přidání x_4	0	−1	1	3	5	4	6	8
přidání x_5	0	2	1	3	5	7	6	8

Vidíme, že optimální hodnota je 8. Optimální řešení dopočítáme zpětně, nebo je možnost v průběhu algoritmu pamatovat jednotlivá dílčí řešení. Optimální řešení jsou dvě, $(0,1,1,0,0)^T$ a $(0,1,0,1,1)^T$.

Poznamenejme, že z výsledné tabulky vidíme také optimální hodnoty pro problémy batohu s pravou stranou postupně $0, 1, \ldots, b$. Nerovnost v zadání problému se nemusí nabýt jako rovnost pro optimální řešení. To je názorně vidět na případu kdy by pravá strana byla $b' = 2$. Pak optimální hodnota není 1, což je maximální hodnota pro řešení splňující nerovnost jako rovnost, ale 2 pro řešení $x = (0,0,0,0,1)^T$, které splňuje nerovnost ostře. □

Přestože je problém batohu speciální úlohou celočíselného programování, lze každý rovnicový 0-1 lineární program zredukovat na problém batohu, varianta (7.3). Této redukci se říká agregace omezení a je popsána dole. Jednoduše řečeno, rovnice dáme dohromady z pohledu q-ární číselné soustavy, kde q je dost velké.

Věta 7.7 (Glover & Woolsey, 1972). *Buď $A \in \mathbb{Z}^{m\times n}$, $b \in \mathbb{Z}^m$ a definujme $q \coloneqq 1 + 2n \max_{ij}\{|a_{ij}|, |b_i|\}$. Pak soustava*

$$Ax = b, \quad x \in \{0,1\}^n \tag{7.4}$$

je ekvivalentní s

$$\sum_{j=1}^{n}\left(\sum_{i=1}^{m} q^{i-1}a_{ij}\right)x_j = \sum_{i=1}^{m} q^{i-1}b_i, \quad x \in \{0,1\}^n. \tag{7.5}$$

Důkaz. „⇒“ Jasné, neboť rovnice (7.5) je jen lineární kombinací rovnic z (7.4).

„⇐“ Buď $k \in \{1,\dots,m\}$. Vezměme obě strany rovnice (7.5) modulo q^k. Bude to

$$\sum_{j=1}^{n}\left(\sum_{i=1}^{k} q^{i-1}a_{ij}\right)x_j = \sum_{i=1}^{k} q^{i-1}b_i, \tag{7.6}$$

neboť pro pravou stranu platí

$$\left|\sum_{i=1}^{k} q^{i-1}b_i\right| \le \sum_{i=1}^{k} q^{i-1}|b_i| \le \frac{1}{2}\sum_{i=1}^{k} q^{i-1}(q-1) = \frac{1}{2}(q^k-1) < \frac{1}{2}q^k,$$

a podobně

$$\left|\sum_{j=1}^{n}\left(\sum_{i=1}^{k} q^{i-1}a_{ij}\right)x_j\right| = \left|\sum_{i=1}^{k} q^{i-1}\left(\sum_{j=1}^{n} a_{ij}x_j\right)\right| \le \frac{1}{2}\sum_{i=1}^{k} q^{i-1}(q-1) < \frac{1}{2}q^k.$$

Protože obě strany rovnice (7.5) mohou být záporné, byl nutný odhad obou stran rovnice (7.6), aby v absolutní hodnotě byly menší než $\frac{1}{2}q^k$.

Podobně pokud vezmeme obě strany rovnice (7.5) modulo q^{k-1}, dostaneme

$$\sum_{j=1}^{n}\left(\sum_{i=1}^{k-1} q^{i-1}a_{ij}\right)x_j = \sum_{i=1}^{k-1} q^{i-1}b_i.$$

Odečtením od rovnice (7.6) pak získáme q^k-násobek k-té rovnice ze soustavy (7.4). Postupnou volbou $k \in \{1,\dots,m\}$ tak vygenerujeme všechny rovnice z (7.4). □

Důsledek 7.8 (Papadimitriou, 1981). *Buď $A \in \mathbb{Z}^{m\times n}$, $b \in \mathbb{Z}^m$. Při pevném m je úloha*

$$\max\ c^T x \quad \textit{za podm.}\quad Ax = b, \quad x \in \{0,1\}^n$$

řešitelná v pseudopolynomiálním čase.

Složitost pseudopolynomiálního algoritmu 7.4 je $\mathcal{O}(nb)$ aritmetických operací, což vzhledem k Preprocessingu I. je také $\mathcal{O}(n\sum_{i=1}^{n} a_i)$. Podobně můžeme sestrojit pseudopolynomiální algoritmus, který bude polynomiální vzhledem k n a koeficientům ve vektoru c.

Algoritmus 7.9 (Pseudopolynomiální algoritmus II.). Uvažujme úlohu s parametrem φ

$$\min\ a^T x \ \text{ za podm. } \ c^T x \ge \varphi, \quad x \in \{0,1\}^n. \tag{7.7}$$

Pokud je optimální hodnota menší nebo rovna b, optimální řešení je přípustným řešením původní úlohy (7.1), mající hodnotu účelové funkce aspoň φ. Jinak takové přípustné řešení neexistuje. Binárním prohledáváním parametru $\varphi \in [0, K]$, kde $K = \sum_{i=1}^{n} c_i$ tak v řádově $\log(K)$ iteracích najdeme optimum (7.1).

Úloha (7.7) je vlastně problém batohu (stačí uvažovat substituci $x' \coloneqq e - x$), lze řešit pseudopolynomiálním algoritmem 7.4 v čase $\mathcal{O}(nK)$. Celkem dostáváme složitost $\mathcal{O}(nK\log(K))$ aritmetických operací.

Pro problém batohu dokonce existuje úplné polynomiální aproximační schema.

Věta 7.10 (Ibarra & Kim, 1975)**.** *V polynomiálním čase vzhledem k $1/\varepsilon$ a velikosti vstupu lze najít přípustné $x^* \in \{0,1\}^n$ tak, že*

$$c^T x^* \geq (1-\varepsilon) \max\ c^T x \quad \textit{za podm.}\ \ a^T x \leq b,\ x \in \{0,1\}^n.$$

Důkaz. Pro $\delta := \varepsilon n^{-1} \max_{i=1,\ldots,n} c_i$ přeškálujeme koeficienty účelové funkce $c'_i := \lfloor c_i/\delta \rfloor$. Díky volbě δ se optimální hodnota změní pouze málo. Protože $c'_i \leq n/\varepsilon$, tak pseudopolynomiální algoritmus 7.9 vyřeší novou úlohu v čase polynomiálním vzhledem k n a $1/\varepsilon$.

Zaokrouhlení v definici c'_i způsobí, že transformace není ekvivalentní a optimum se může změnit. Na druhou stranu, díky volbě δ se optimální hodnota změní pouze málo. Je-li x^* optimum úlohy s účelovou funkcí c', a x^{opt} optimum původní úlohy, tak

$$\begin{aligned} c^T x^* &\geq \delta c'^T x^* \geq \delta c'^T x^{\text{opt}} = \delta \lfloor c^T/\delta \rfloor x^{\text{opt}} \geq \delta (c/\delta - e)^T x^{\text{opt}} \\ &\geq c^T x^{\text{opt}} - \delta n = c^T x^{\text{opt}} - \varepsilon \max_i c_i \geq (1-\varepsilon) c^T x^{\text{opt}}, \end{aligned}$$

kde jsme v poslední nerovnici použili fakt, že $c^T x^{\text{opt}} \geq \max_i c_i$ díky Předpokladu 2. □

7.2 Relaxace

Relaxace pro (7.1)

Relaxace (7.1) vede na lineární program

$$\max\ c^T x \ \text{ za podm. } \ a^T x \leq b,\ 0 \leq x \leq 1,$$

jehož řešení x^R můžeme vyjádřit explicitně. Předpokládejme, že proměnné jsou setříděné tak, že $\frac{c_1}{a_1} \geq \ldots \geq \frac{c_n}{a_n}$. Pak x^R je ve tvaru

$$x_1^R = \ldots = x_{r-1}^R = 1,\ x_r^R = \frac{b - \sum_{i=1}^{r-1} a_i}{a_r},\ x_{r+1}^R = \ldots = x_n^R = 0$$

pro určité $r \in \{1, \ldots, n\}$. Optimální hodnota je pak

$$c^T x^R = \sum_{i=1}^{r-1} c_i + \frac{b - \sum_{i=1}^{r-1} a_i}{a_r} c_r.$$

Optimální řešení nám rovněž nabízí heuristické přípustné řešení x^H ve tvaru

$$x_1^H = \ldots = x_{r-1}^H = 1,\ x_r^H = \ldots = x_n^H = 0$$

a účelovou hodnotou $c^T x^H = \sum_{i=1}^{r-1} c_i$. Optimální hodnotu tedy můžeme těsně zapouzdřit do intervalu

$$[c^T x^H,\ c^T x^R] = \left[\textstyle\sum_{i=1}^{r-1} c_i,\ \sum_{i=1}^{r-1} c_i + \frac{b - \sum_{i=1}^{r-1} a_i}{a_r} c_r \right].$$

Poznámka 7.11. Setřídění posloupnosti $\frac{c_1}{a_1}, \ldots, \frac{c_n}{a_n}$ vyžaduje čas $\mathcal{O}(n \log n)$. Relaxované optimum můžeme najít dokonce v lineárním čase bez třídění. Využijeme toho, že medián posloupnosti lze najít v $\mathcal{O}(n)$. Pokryjeme-li kapacitu b pomocí první půlky, zaměříme se rekursivně na ni, jinak proměnné první půlky nastavíme na 1 a rekursivně se zaměříme na druhou půlku. Tak potřebujeme pouze čas $\mathcal{O}(n) + \mathcal{O}(n/2) + \mathcal{O}(n/4) + \ldots = \mathcal{O}(n)$.

Relaxace pro (7.2)

Podobně můžeme postupovat pro variantu (7.2) problému batohu. Relaxovaná úloha má optimální řešení $x^R = (\frac{b}{a_1}, 0, \ldots, 0)^T$ s optimální hodnotou $\frac{b}{a_1}c_1$. Heuristické přípustné řešení jest $x^R = (\lfloor\frac{b}{a_1}\rfloor, 0, \ldots, 0)^T$ s účelovou hodnotou $\lfloor\frac{b}{a_1}\rfloor c_1$. Pro tuto variantu umíme dokázat i přesnost aproximace.

Tvrzení 7.12. *Heuristické řešení není víc jak dvakrát horší než optimum, tj.* $c^T x^H \geq \frac{1}{2}c^T x^*$.

Důkaz. Ukážeme víc, dokonce $c^T x^H \geq \frac{1}{2}c^T x^R$. Z Preprocessingu I. víme, že $a_1 \leq b$, tedy $\lfloor\frac{b}{a_1}\rfloor \geq 1 \geq \{\frac{b}{a_1}\}$. Nyní

$$\frac{c^T x^H}{c^T x^R} = \frac{\lfloor\frac{b}{a_1}\rfloor c_1}{\frac{b}{a_1}c_1} = \frac{\lfloor\frac{b}{a_1}\rfloor}{\lfloor\frac{b}{a_1}\rfloor + \{\frac{b}{a_1}\}} \geq \frac{\lfloor\frac{b}{a_1}\rfloor}{2\lfloor\frac{b}{a_1}\rfloor} = \frac{1}{2}. \qquad \square$$

Příklad 7.13. Pro variantu (7.1) takováto relativní chyba neplatí. Uvažujme například úlohu s $a = (1, 100)^T$, $b = 100$, $c = (2, 100)^T$. Pak heuristické řešení je $x^H = (1, 0)^T$ s hodnotou účelové funkce $c^T x^H = 2$, ale optimální řešení je $x^* = (0, 1)^T$ s optimální hodnotou $c^T x^* = 100$. $\square$

Odhad absolutní chyby:

Tvrzení 7.14. *Pro obě varianty (7.1) a (7.2) je* $c^T x^* - c^T x^H \leq c^T x^R - c^T x^H \leq \max_{i=1,\ldots,n} c_i$.

Důkaz. Pro (7.1) je $c^T x^R - c^T x^H \leq c_r \leq \max_i c_i$. Pro (7.2) je $c^T x^R - c^T x^H \leq c_1 \leq \max_i c_i$. $\square$

7.3 Sečné nadroviny

Budeme uvažovat sečné nadroviny tzv. typu „cover inequalities“. Buď $C \subseteq \{1, \ldots, n\}$ tak, že $\sum_{i\in C} a_i > b$. Pak nemohou být všechny proměnné z C nastaveny na jedničku, a proto lze přidat podmínku

$$\sum_{i\in C} x_i \leq |C| - 1. \tag{7.8}$$

Množina C se nazývá *cover*, a speciálně je to minimální cover, pokud žádná vlastní podmnožina už cover netvoří.

Podmínku (7.8) lze jednoduše zesílit na tvar

$$\sum_{i\in J} x_i \leq |C| - 1,$$

kde $J \coloneqq C \cup \{j;\, a_j \geq a_i\ \forall i \in C\}$.

Jiný způsob jak zesílit podmínku je tzv. *lifting*. Cílem je dostat podmínku tvaru

$$\sum_{i\notin C} \alpha_i x_i + \sum_{i\in C} x_i \leq |C| - 1,$$

kde $\alpha_i \in \mathbb{N}$. Pro jednoduchost předpokládejme, že $C = \{s+1, \ldots, n\}$. Hodnoty $\alpha_1, \ldots, \alpha_s$ budeme počítat postupně, V iteraci t již známe $\alpha_1, \ldots, \alpha_{t-1}$ chceme určit co největší $\alpha_t \in \mathbb{N}$ aby platilo

$$\alpha_t x_t + \sum_{i=1}^{t-1} \alpha_i x_i + \sum_{i\in C} x_i \leq |C| - 1$$

pro všechna přípustná x, což vede na úlohu

$$\alpha_t = |C| - 1 - \max\left\{ \sum_{i=1}^{t-1} \alpha_i x_i + \sum_{i\in C} x_i;\ \ \sum_{i=1}^{t-1} a_i x_i + \sum_{i\in C} a_i x_i \leq b - a_t,\ x \in \{0,1\}^{n-s+t-1} \right\}.$$

Toto je sice opět problém batohu, ale menší dimenze, což bývá ještě umocněno tím, že často bývá $\alpha_i = 0$. Pokud tuto úlohu vyřešíme přesně, je výsledná sečná nadrovina stěnová [Wolsey, 1998], nicméně i dolní odhad na α_t může dát silný řez.

Zbývá otázka jak volit cover C. To vyplyne často samo. Je-li x^* neceločíselné optimální řešení relaxace, pak můžeme volit $C \coloneqq \{i;\, x_i^* > 0\}$.

Příklad 7.15. Uvažme příklad 7.6 a buď x^* optimum relaxace. Najděte příslušný minimální cover a pomocí liftingu zesilte řez. Přidejte řez do podmínek a najděte příslušné relaxované optimální řešení.

Řešení: Setříděním proměnných sestupně podle c_i/a_i dostaneme úlohu

$$\max\ 2x_1 + 5x_2 + 3x_3 + x_4 + x_5 \ \text{ za podm. } \ x_1 + 4x_2 + 3x_3 + 2x_4 + 3x_5 \leq 7,\ x \in \{0,1\}^5.$$

Lineární relaxace má optimální řešení $x^R = (1, 1, \frac{2}{3}, 0, 0)^T$ a příslušný cover je $C = \{1, 2, 3\}$. Snadno se nahlédne, že je minimální.

Lifting α_4: Spočítáme úlohu

$$\max\ x_1 + x_2 + x_3 \ \text{ za podm. } \ x_1 + 4x_2 + 3x_3 \leq 7 - a_4(= 5),\ x \in \{0,1\}^5.$$

Optimální hodnota je 2, tudíž $\alpha_4 = |C| - 1 - 2 = 0$.

Lifting α_5: Spočítáme úlohu

$$\max\ x_1 + x_2 + x_3 \ \text{ za podm. } \ x_1 + 4x_2 + 3x_3 \leq 7 - a_5(= 4),\ x \in \{0,1\}^5.$$

Optimální hodnota je 2, tudíž $\alpha_5 = |C| - 1 - 2 = 0$.

Tudíž nejsilnější cover nerovnost je $x_1 + x_2 + x_3 \leq 2$. Po přidání k relaxované úloze dostaneme lineární program

$$\begin{aligned} \max\ & 2x_1 + 5x_2 + 3x_3 + x_4 + x_5 \\ \text{za podm. } & x_1 + 4x_2 + 3x_3 + 2x_4 + 3x_5 \leq 7,\ x_1 + x_2 + x_3 \leq 2,\ x \in [0,1]^5, \end{aligned}$$

jehož optimální řešení $(0, 1, 1, 0, 0)^T$ je již celočíselné. □

7.4 Branch & cut

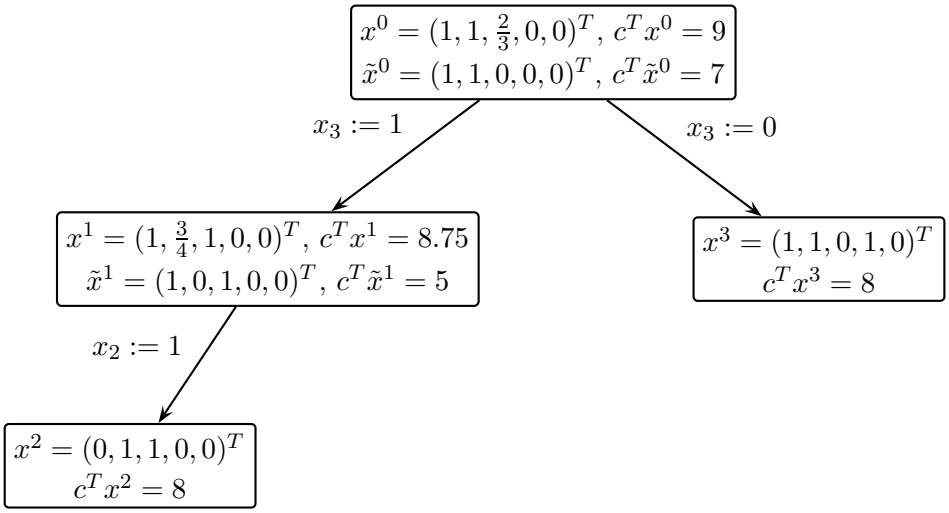
Metoda branch & bound má pro problém batohu jednodušší podobu. S každým rozvětvením klesne dimenze podproblému o 1 (tak jako pro každý 0-1 celočíselný program) a navíc z předchozího víme, že relaxovanou úlohu v každém uzlu umíme hned vyřešit. Zde se významně uplatní Preprocessing I.

Nicméně, jak jsme viděli v příkladu 3.5, i pro snadnou úlohu musí někdy metoda branch & bound provést exponenciální počet kroků. Přidáním sečných nadrovin sice ztratíme explicitní vyjádření řešení relaxovaných podúloh, zesílíme tím ale tyto relaxace a výpočetní čas většinou zlepšíme.

Podle Schrijver [1998] experimenty prováděné Balasem a Zemelem s náhodným vstupem a 10^4 proměnnými trvaly už roce 1980 pouze ca jednu sekundu. Použitá metoda kombinovala branch & bound, sečné nadroviny a dynamické programování.

Příklad 7.16. Vyřešte příklad 7.6 pomocí základní metody branch & bound.

Řešení: Postup metody je znázorněný v následujícím stromu:



V kořeni spočítáme řešení relaxované úlohy $x^0 = (1, 1, \frac{2}{3}, 0, 0)^T$ a příslušnou horní mez na optimální hodnotu $c^T x^0 = 9$. Odpovídající heuristické přípustné řešení je $\tilde{x}^0 = (1, 1, 0, 0, 0)^T$ a účelová hodnota $c^T \tilde{x}^0 = 7$. V levé větvi zafixujeme $x_3 := 1$. Optimum relaxované úlohy je $x^1 = (1, \frac{3}{4}, 1, 0, 0)^T$ s hodnotou $c^T x^1 = 8.75$. Příslušné heuristické řešení je $\tilde{x}^1 = (1, 0, 1, 0, 0)^T$, ale účelová hodnota $c^T \tilde{x}^1 = 5$ je malá, takže jej nemusíme ukládat. Další levá větev má optimum relaxace rovnou celočíselné $x^2 = (0, 1, 1, 0, 0)^T$. Účelová hodnota je $c^T x^2 = 8$, což vylepšuje dosavadní nejlepší přípustné řešení. Zároveň to dává lepší dolní odhad na optimální hodnotu. Tudíž backtrackujeme zpět: otcovský uzel můžeme vynechat, protože horní mez $c^T x^1 = 8.75$ nic lepšího neslibuje. Dojedeme až ke kořeni a vydáme se pravou větví s omezením $x_3 = 0$. Optimum relaxované úlohy je $x^3 = (1, 1, 0, 1, 0)^T$ s hodnotou $c^T x^3 = 8$. Tudíž 8 musí být optimální hodnota a x^2 a x^3 dvě optimální řešení. □

7.5 Preprocessing II.

Buď r hodnota ze sekce věnované relaxaci a předpokládejme i setříděnost proměnných. Tím pádem se bude čekat, že optimální řešení pro první proměnné budou spíš jedničky, a pro ty na konci spíš nuly. Proto postupně pro každé $i = 1, \ldots, r$ zafixujeme $x_i = 0$, a pokud horní mez optimální hodnoty úlohy s touto fixací (spočítaná např. relaxací) bude menší nebo rovna dolní mezi optimální hodnoty původní úlohy (spočítané např. heuristikou), pak můžeme natrvalo přiřadit $x_i = 1$. Podobně, pro každé $i = r + 1, \ldots, n$ zafixujeme $x_i = 1$, a pokud horní mez optimální hodnoty úlohy s touto fixací bude menší nebo rovna dolní mezi optimální hodnoty původní úlohy , pak natrvalo přiřadíme $x_i = 0$.

Abychom nemuseli počítat v každém kroku znovu relaxované optimum (přestože to nestojí moc práce), můžeme optimální hodnotu relaxace po zafixování $x_i = 0$, $i < r$, shora odhadnout pomocí $c^T x^R - c_i + \frac{c_r}{a_r} a_i$, což zabere konstantní čas. Podobně po zafixování $x_i = 1$, $i > r$, použijeme horní odhad $c^T x^R + c_i - \frac{c_r}{a_r} a_i$.

7.6 Aplikace: sečné nadroviny pro 0-1 programování

Uvažujme 0-1 program

$$\max\ c^T x \ \text{ za podm. } \ Ax \leq d, \ \ x \in \{0,1\}^n,$$

kde $A \in \mathbb{Z}^{m \times n}$, $c \in \mathbb{Z}^n$ a $d \in \mathbb{Z}^m$. Je-li matice A velká a řídká, pak jednotlivé nerovnice se chovají téměř jako samostatné problémy batohu. Je-li x^* neceločíselné optimum relaxace, tak můžeme řezné nadroviny vytvářet pomocí cover nerovnic pro jednotlivé nerovnice soustavy. Tento postup se prý osvědčil [Schrijver, 1998].

Uvažujme jen jednu z nerovnic ve tvaru $a^T x \leq b$ a chceme najít cover $C \subseteq \{1, \ldots, n\}$ tak, aby

$$\sum_{i \in C} a_i > b, \quad \sum_{i \in C} x_i^* > |C| - 1,$$

nebo spíše

$$\sum_{i \in C} a_i \geq b + 1, \quad \sum_{i \in C} x_i^* \geq |C|. \tag{7.9}$$

První podmínka vychází z definice coveru a druhá slouží k odříznutí bodu x^* sečnou nadrovinou (7.8). Nyní nefunguje postup ze sekce 7.3 na vytvoření C, protože cover definovaný jako $C := \{i;\ x_i^* > 0\}$ by nemusel splňovat podmínku $\sum_{i \in C} a_i > b$, a musíme na to tudíž jinak. Zavedeme-li $z \in \{0,1\}^n$ indikátorové proměnné množiny C, tak hledáme řešení

$$\sum_{i=1}^{n} a_i z_i \geq b + 1, \quad \sum_{i=1}^{n} x_i^* z_i \geq \sum_{i=1}^{n} z_i,$$

což vede na problém batohu

$$\min\ \sum_{i=1}^{n} (1 - x_i^*) z_i \ \text{ za podm. } \ \sum_{i=1}^{n} a_i z_i \geq b + 1, \ z \in \{0,1\}^n. \tag{7.10}$$

Tvrzení 7.17. *Buď z^* optimální řešení a α^* optimální hodnota (7.10).*

(1) Je-li $\alpha^* > 0$, tak x^* splňuje každou cover nerovnici typu (7.9) pro $a^T x \le b$.

(2) Je-li $\alpha^* \le 0$, tak $C := \{i;\, z_i^* = 1\}$ je cover pro nejvíce porušenou cover nerovnost $\sum_{i\in C} x_i \le |C|-1$ (je porušena o $1-\alpha^*$).

Přestože takovéto generování sečných nadrovin vyžaduje řešit problémy batohu jako podúlohy, je takto strávený čas výrazně menší v porovnání s původní úlohou. Navíc místo přesného řešení lze aplikovat heuristiky (ale na úkor síly řezu) nebo ε-aproximaci dle věty 7.10.

Cvičení

7.1. Uvažujme varianty problému batohu

$$\max\ c^T x \ \text{ za podm. } \ a^T x \le b,\ x \in \{0,1\}^n, \tag{P_1}$$
$$\max\ c^T x \ \text{ za podm. } \ a^T x \le b,\ x \ge 0,\ x \in \mathbb{Z}^n, \tag{P_2}$$
$$\max\ c^T x \ \text{ za podm. } \ Ax \le b,\ x \in \{0,1\}^n, \tag{P_3}$$
$$\max\ c^T x \ \text{ za podm. } \ a^T x = b,\ x \in \{0,1\}^n, \tag{P_4}$$

kde $c, A, a, b \ge 0$.

(a) Ukažte, že předpoklad $a, b, c > 0$ není v (P_2) na újmu obecnosti. Jak je to u (P_3) a (P_4)?

(b) Rozhodněte, která z variant (P_1)–(P_4) je převoditelná na nějakou jinou, popř. které jsou na sebe navzájem převoditelné.

(c) Pro (P_3) navrhněte heuristiku a horní odhad její relativní chyby.

(d) Prostudujte metodu Branch & Bound pro problémy (P_3) a (P_4).

(e) Prostudujte preprocessing pro problémy (P_2)–(P_4).

7.2. Aplikujte Lagrangeovu relaxaci (a porovnejte ji s klasickou celočíselnou relaxací) na problém batohu (7.1).

Kapitola 8

Set covering

Pokrývací úloha *set covering* má tvar

$$\min\ c^T x\ \text{ za podm. }\ Ax \geq e,\ \ x \in \{0,1\}^n, \tag{8.1}$$

kde $A \in \{0,1\}^{m\times n}$ a $c \in \mathbb{N}^n$.

Příklad 8.1. Zformulujte pomocí (8.1) úlohu vrcholového pokrytí grafu a úlohu maximálního párování v grafu.

Řešení (Vrcholové pokrytí): Buď $G = (V, E)$, $|V| = n$, graf a chceme najít nejmenší množinu vrcholů tak, aby z každé hrany obsahovala aspoň jeden konec. Problém vede na formulaci

$$\min\ e^T x\ \text{ za podm. }\ A^T x \geq e,\ \ x \in \{0,1\}^n,$$

kde A je matice incidence grafu G (definice 1.9). Proměnné jsou booleovské indikátory vrcholů, které vybereme, a i-tá nerovnost v omezeních pak říká, že hrana i musí být pokryta.

Řešení (Maximální párování): Buď $G = (V, E)$, $|E| = m$, graf a chceme najít největší množinu disjunktních hran. Problém vede na formulaci

$$\max\ e^T x\ \text{ za podm. }\ Ax \leq e,\ \ x \in \{0,1\}^m,$$

kde A je matice incidence grafu G. Proměnné jsou booleovské indikátory hran, které vybereme, a i-tá nerovnost v omezeních pak říká, že vrchol i musí být pokryt maximálně jednou hranou. Poznámka: Při váženém párování nahradíme účelovou funkci $e^T x$ za $c^T x$.

Nyní zavedeme substituci $y \coloneqq e - x$ a úloha bude

$$m - \min\ e^T y\ \text{ za podm. }\ Ay \geq Ae - e,\ \ y \in \{0,1\}^m.$$

Přestože to není přesně požadovaný tvar (8.1), je to velmi podobný problém. Jediný rozdíl je v pravé straně omezení, kdy vektor $Ae - e$, udávající seznam stupňů vrcholů snížený o 1, se obecně neskládá jen z jedniček. Nicméně za předpokladu, že G neobsahuje vrcholy stupně nula nebo jedna (což je bez újmy na obecnost), je vektor $Ae - e$ nezáporný. □

Příklad 8.2 (Emergency servis centers)**.** Pomocí minimálního počtu nouzových středisek chceme pokrýt celý kraj. Tento problém vede přímo na formulaci (8.1), kde x_j udává jestli se j-té středisko otevře či nikoli a a_{ij} udává zda j-té středisko dokáže obsloužit i-tou oblast. □

Poznámka 8.3 (Set partitioning)**.** Podobná úloha je problému set covering je např. *set partitioning* s rovnostmi namísto nerovností:

$$\min\ c^T x\ \text{ za podm. }\ Ax = e,\ \ x \in \{0,1\}^n,$$

kde $A \in \{0,1\}^{m\times n}$ a $c \in \mathbb{N}^n$. Jak ukazujeme dole, tato úloha lze převést na (8.1), takže se budeme nadále věnovat pouze set coveringu.

Důkaz. Úloha set partitioningu je ekvivalentní s úlohou

$$\min\ c^T x + \alpha e^T y \ \text{ za podm. } \ Ax - y = e, \ \ y \geq 0, \ \ x \in \{0,1\}^n, \ y \in \mathbb{N}^m,$$

kde $\alpha > 0$ je dostatečně velké (stačí brát $\alpha = e^T c$). Eliminací y (tj., dosazením $y \coloneqq Ax - e$) dostaneme úlohu

$$\min\ c^T x + \alpha e^T (Ax - e) \ \text{ za podm. } \ Ax \geq e, \ \ x \in \{0,1\}^n,$$

neboli

$$\alpha m + \min\ (c^T + \alpha e^T A)x \ \text{ za podm. } \ Ax \geq e, \ \ x \in \{0,1\}^n.$$

Pokud je optimální hodnota aspoň α, znamená to, že původní úloha set partitioningu není přípustná. V opačném případě se optimální řešení i hodnota rovnají. □

8.1 Branch & bound

Metodu branch & bound pro úlohu set coveringu prvně zkoumali Lemke, Salkin & Spielberg (1971).

Metoda branch & bound v principu funguje tak, jak jsme ji popisovali v kapitole 3, ale má přeci jenom jistá specifika.

Pokud větvíme úlohu na dvě podúlohy podle proměnné x_k, tak obě podúlohu mají menší dimenzi díky přidané podmínce $x_k = 1$ resp. $x_k = 0$. V prvním případě se navíc sníží počet omezení, protože pro $a_{ik} = 1$ se stává i-tá nerovnice redundantní. Tudíž můžeme vypustit $e^T A_{*k}$ nerovnic.

Jak volit k podle kterého se větvíme? Existuje několik přístupů:

- $k \coloneqq \arg\min_j c_j$, protože tato proměnná má nejmenší vliv na účelovou funkci.
- $k \coloneqq \arg\max_j e^T A_{*j}$, protože čím více jedniček má matice A v k-tém sloupci, tím více nerovností pak bude redundantních v jedné větvi prohledávacího stromu a rychleji najdeme dobré řešení.
- $k \coloneqq \arg\min_j \frac{c_j}{e^T A_{*j}}$, kombinace předchozích dvou přístupů.
- Typicky se volba ještě zúžuje na tzv. *preferable set* P, což jsou indexy jedniček v tom řádku matice A s nejméně jedničkami. Protože aspoň jedna z nich musí být pokryta, je větší šance, že větev stromu s $x_k = 1$ vede k optimu.

8.2 Preprocessing

Několik jednoduchých pravidel, které se projeví zejména během procházení výpočetního stromu metodou branch & bound:

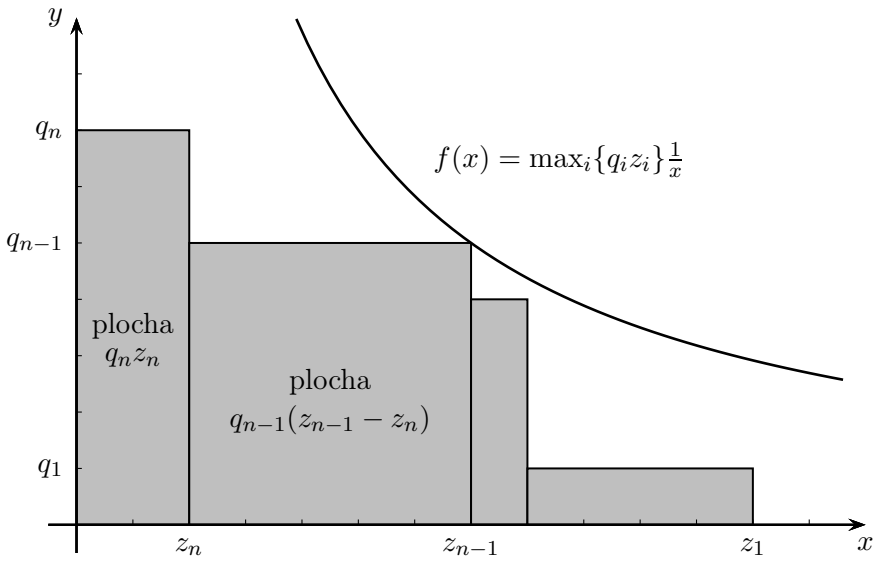
- Existuje-li řádek matice A se samými nulami, tj. $A_{i*} = 0^T$, pak je úloha nepřípustná (a naopak).
- Existuje-li řádek matice A s právě jednou jedničkou na k-té pozici, tj. $A_{i*} = e_k^T$, pak můžeme zafixovat $x_k \coloneqq 1$.
- Řádková dominance: Je-li $A_{i*} \geq A_{j*}$, pak i-tá nerovnost je redundantní a lze vypustit.
- Sloupcová dominance: Pokud pro nějakou indexovou množinu $J \subseteq \{1, \ldots, n\} \setminus \{k\}$ platí $\sum_{j \in J} A_{*j} \geq A_{*k}$ a $\sum_{j \in J} c_j \leq c_k$, pak lze zafixovat $x_k \coloneqq 0$.

8.3 Hladový algoritmus

Algoritmus 8.4 (Hladový algoritmus pro set covering problem)**.**

1: Polož $M \coloneqq \{1, \ldots, m\}$, $x^H = 0$ a $S_j \coloneqq \{i;\, a_{ij} = 1\}$.

2: Dokud $M \neq \emptyset$ dělej:



Obrázek 8.1: Ilustrace k důkazu Lemmatu 8.5.

- $j^* \coloneqq \arg\min_{j:\, x_j^H = 0} \frac{c_j}{|M \cap S_j|}$,
- $x_{j^*}^H \coloneqq 1$,
- $M \coloneqq M \setminus S_{j^*}$.

Množina M udává indexy ještě nepokrytých nerovností, volba j^* odpovídá heuristice ze sekce 8.1.

Lemma 8.5. *Buďte $0 < q_1 \leq \ldots \leq q_n$ reálná a $z_1 \geq \ldots \geq z_n$ přirozená. Pak platí*

$$\sum_{i=1}^{n-1} q_i(z_i - z_{i+1}) + q_n z_n \leq \max_{i=1,\ldots,n}\{q_i z_i\} H(z_1), \tag{8.2}$$

kde $H(k) = 1 + 1/2 + 1/3 + \ldots + 1/k$.

Důkaz. Levá strana nerovnosti (8.2) representuje součet obsahů obdélníků $(0, z_n) \times (0, q_n)$, $(z_n, z_{n-1}) \times (0, q_{n-1})$, $\ldots$, $(z_2, z_1) \times (0, q_1)$, viz obrázek 8.1. Graf funkce $f(x) = \max_i\{q_i z_i\} x^{-1}$ leží nad těmito obdélníky, což se snadno ověří dosazením hodnot $z_n, \ldots, z_1$: Pro $x = z_k$ obdélník zasahuje vysoko q_k, zatímco $f(z_k) = \max_i\{q_i z_i\} z_k^{-1} \geq q_k z_k z_k^{-1} = q_k$.

Součet obsahů obdélníků shora odhadneme plochou pod grafem $f(x)$, přesněji řešeno plochu pod grafem pouze odhadneme součtem funkčního hodnot v bodech $1, \ldots, z_1$ (protože $z_n, \ldots, z_1 \in \mathbb{Z}$, a tudíž dostaneme jemnější dělení). To dá přímo pravou stranu nerovnosti (8.2). □

Věta 8.6. *Buď x^* optimální řešení (8.1). Pak $c^T x^H \leq H(d) c^T x^*$, kde $d \coloneqq \max_{j=1,\ldots,n} |S_j|$.*

Důkaz. Uvažujme relaxovanou úlohu k (8.1) ve tvaru lineárního programu

$$\min\ c^T x \ \text{ za podm. } \ Ax \geq e, \ \ x \geq 0. \tag{8.3}$$

Duální problém je

$$\max\ e^T u \ \text{ za podm. } \ A^T u \leq c, \ \ u \geq 0. \tag{8.4}$$

Namísto požadovaného odhadu ve skutečnosti dokážeme více, a to

$$c^T x^H = H(d) \cdot e^T u \leq H(d) \cdot c^T x^R \leq H(d) \cdot c^T x^*,$$

kde x^R je optimální řešení relaxace (8.3) a u je jisté přípustné řešení duálního problému (8.4).

Označme jako T počet iterací hladového algoritmu a pro iteraci t nechť M_t značí aktuální množinu M indexů nepokrytých nerovnic. Dále označme

$$q_t \coloneqq \frac{c_{j_t^*}}{|M_t \cap S_{j_t^*}|} = \min_{j\colon x_j^H = 0} \frac{c_j}{|M_t \cap S_j|}.$$

Nyní definujme přípustné řešení duálního problému (8.4) jako

$$u_i = \frac{q_t}{H(d)} \quad \text{pro } i \in M_t \setminus M_{t+1}.$$

Takto definované u je duálně přípustné, neboť pro každé $j \in \{1, \ldots, n\}$ je

$$\begin{aligned}(A^T u)_j &= \sum_{i=1}^{m} u_i a_{ij} = \sum_{t=1}^{T} \frac{q_t}{H(d)} \sum_{i \in M_t \setminus M_{t+1}} a_{ij} = \sum_{t=1}^{T} \frac{q_t}{H(d)} \left(\sum_{i \in M_t} a_{ij} - \sum_{i \in M_{t+1}} a_{ij} \right) \\ &= \frac{1}{H(d)} \sum_{t=1}^{T} q_t (|M_t \cap S_j| - |M_{t+1} \cap S_j|).\end{aligned}$$

Podle Lemmatu 8.5 s volbou q_t a $z_t \coloneqq |M_t \cap S_j|$ můžeme tuto hodnotu shora odhadnout

$$\begin{aligned}(A^T u)_j &\le \frac{1}{H(d)} \max_{t=1,\ldots,T} \{q_t |M_t \cap S_j|\} \cdot H(|M_1 \cap S_j|) \\ &\le \frac{1}{H(d)} \max_{t=1,\ldots,T} \left\{ \frac{c_j}{|M_t \cap S_j|} |M_t \cap S_j| \right\} \cdot H(d) = c_j.\end{aligned}$$

Nakonec porovnáme hodnoty duální účelové funkce a hladového řešení:

$$e^T u = \frac{1}{H(d)} \sum_{t=1}^{T} q_t (|M_t| - |M_{t+1}|) = \frac{1}{H(d)} \sum_{t=1}^{T} c_{j_t^*} = \frac{1}{H(d)} c^T x^H. \qquad \square$$

Hladový algoritmus tedy dává řešení, která nejsou víc než $H(d)$-krát horší než optimum. Navíc, poměrovou chybu můžeme odhadnout $H(d) \le H(m) \approx \log(m)$. Celkem se věří, že neexistuje řádově lepší polynomiální aproximační algoritmus, protože by to pak znamenalo silné důsledky pro třídu NP [Feige, 1998].

Cvičení

8.1. Pomocí branch & bound metody vyřešte úlohu set partitioningu a daty

$$c = (3, 2, 1, 1, 3, 2)^T, \quad A = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

8.2. Pomocí branch & bound metody vyřešte úlohu set coveringu a daty

$$c = (4, 4, 4, 8, 3, 6)^T, \quad A = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

8.3. Najděte příklad, kdy se nabyde (co nejtěsněji) mez z věty 8.6.

Kapitola 9

Problém obchodního cestujícího

V problému obchodního cestujícího (TSP, traveling salesman problem) chceme najít nejkratší cestu procházející n městy tak, abychom každé město navštívili právě jednou. Tomuto populárnímu problému se věnuje řada prací jako Applegate et al. [2006]; Cook [2012].

Matematická formulace úlohy: Dán úplný orientovaný graf s vrcholy $V = \{1, \dots, n\}$ a matice ohodnocení $C \in \mathbb{R}^{n \times n}$, kde $c_{ii} = \infty \ \forall i$. Pak řešíme optimalizační úlohu

$$\min \sum_{i,j:i\neq j} c_{ij}x_{ij} \quad \text{za podm.} \quad \sum_{i:\, i\neq k} x_{ik} = \sum_{j:\, j\neq i} x_{kj} = 1 \ \forall k \in V, \tag{9.1a}$$

$$x_{ij} \in \{0,1\} \ \forall i \neq j, \tag{9.1b}$$

$$x \text{ představuje hamiltonovskou kružnici.} \tag{9.1c}$$

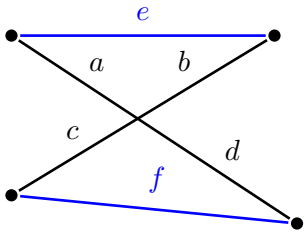
Rozeznáváme různé typy TSP. *Symetrické TSP* má matici C symetrickou a úloha se zjednodušuje na

$$\min \sum_{e\in E} c_e x_e \quad \text{za podm.} \quad \sum_{e\ni k} x_e = 2 \ \forall k \in V, \ x_e \in \{0,1\} \ \forall e \in E, \ \text{(9.1c)}.$$

Pokud navíc platí trojúhelníková nerovnost $c_{ij} \leq c_{ik} + c_{kj}$ pro všechna i, j, k, $i \neq j$, tak se jedná o *metrické TSP*. A nejspeciálnějším typem je *eukleidovské TSP*, pro něž existuje vnoření do roviny tak, že vrcholy odpovídají bodům a c_{ij} eukleidovským vzdálenostem mezi nimi.

Tvrzení 9.1. *V eukleidovském TSP existuje optimální cesta a nakreslení v rovině s nekřížícími se hranami.*

Důkaz. Uvažujme dvojici křížících se hran, vyznačených černě na obrázku:



Tvrdíme, že tyto dvě hrany lze nahradit dvěma vodorovnými, vyznačenými modře (Případně dvěma svislými, podle toho, která z těchto dvou variant vede na hamiltonovskou kružnici. Ta druhá varianta vede na dvě podkružnice.) Z trojúhelníkové nerovnosti máme $e \leq a + b$, $f \leq c + d$, z čehož součtem je $e + f \leq (a + d) + (b + c)$. Tudíž součet délek nových hran je nanejvýš rovno součtu délek původních hran. Nejkratší cestu proto neprodloužíme. □

Tvrzení 9.2. *Optimální cesta v eukleidovském TSP prochází body na hranici konvexního obalu bodů v tom pořadí, ve kterém jsou uspořádány podél této hranice.*

Příklad 9.3. Použití TSP v praxi:

1. doprava & logistika (rozvážení zboží zákazníkům, ...),

2. Výroba a testování čipů: Při ověřování funkčnosti hotových čipů se spojují jednotlivé komponenty do cest, na kterých se provádí testování zkušebními daty. TSP umožňuje minimalizovat cesty, a tak i zkrátit testovací fázi [Cook, 2012].
3. Videohry: Při načítání textur ke scénám je třeba načítat z disku z různých míst. Aby se zefektivnila rychlost načítání, má smysl poskládat data cíleně, např. podle nejlepší cesty v grafu, kde vrcholy odpovídají texturám a délka hrany počtu scén, které obsahují jednu texturu, ale ne druhou [Cook, 2012].
4. Psychologie: Test se skládá z nalezení TSP s vrcholy označenými 1, A, 2, B, ..., 12, L, 13. Existuje souvislost s úspěšností v testu a poškozením mozku. Průzkum v r. 1990 ukázal, že je nejrozšířenějším testem a je stále stejný, aby se dosahovalo statisticky relevantních výsledků [Cook, 2012].
5. Archeologie: (relativní) datování hrobů. Petrie už v roce 1899 využil něco jako TSP k nalezení časové souslednosti hrobů v Horním Egyptě. Vrcholy TSP odpovídají hrobům a vzdálenosti pak Hammingově vzdálenosti určitých rysů a artefaktů nalezených v hrobech [Grötschel, 2012, str. 199].
6. řazení genů aj. □

Definice 9.4 (Aproximační faktor)**.** Říkáme, že algoritmus řeší TSP s aproximačním faktorem φ pokud dává přípustné řešení, které není víc jak φ-krát horší než optimum.

Poznámka 9.5 (Složitost)**.** TSP je NP-těžký problém, protože pomocí něho dokážeme vyřešit úlohu existence hamiltonovské kružnice v grafu. Toto platí i pokud se omezíme na bipartitní grafy, dokonce i jen na grafy ve tvaru podmnožiny 2D mřížky (Itai, Papadimitriou & Szwarcfiter).

NP-těžkost zůstane zachována i pro eukleidovské TSP, stejně jako pro různé modifikace TSP jako např. nalezení nejdelší cesty, bottleneck TSP (nalezení cesty, jejíž nejdelší hrana je co nejmenší) aj. Další složitostní výsledky:

- [Papadimitriou and Vempala, 2006]
  Je NP-těžké aproximovat nesymetrické metrické TSP s faktorem $\frac{117}{116}$.
- [Papadimitriou and Vempala, 2006]
  Je NP-těžké aproximovat symetrické metrické TSP s faktorem $\frac{220}{219}$.
- [Arora, 1996; Mitchell, 1999]
  Existuje polynomiální aproximační schema pro eukleidovské TSP.

Zatím je otevřená otázka jestli existuje aproximační algoritmus na metrické TSP s konstantním faktorem menším než 1.5 (srov. Christofidův algoritmus dole). Pro obecné TSP žádný takový algoritmus neexistuje. Otevřené je také jestli eukleidovské TSP je ve třídě NP, protože díky eukleidovské metrice musíme počítat s odmocninami.[1)]

Tvrzení 9.6. *Pro obecné TSP neexistuje polynomiální algoritmus s konstantním aproximačním faktorem.*

Důkaz. Redukcí z problému hledání hamiltonovské kružnice grafu $G = (V, E)$. Hrany ohodnoť 1, a chybějící hrany doplň s váhou $1 + r$, kde $r > 0$. Pokud původní graf G obsahuje hamiltonovskou kružnici, tak optimální hodnota TSP je n, jinak aspoň $n + r$. Poměr je tedy $1 + r/n$ a může být libovolně velký. □

9.1 Formulace

Podmínku (9.1c) na hamiltonovskou kružnici lze zformulovat několika způsoby

(1) *podmínka Miller–Tucker–Zemlin (1960):* přidáme proměnné v_i, $i = 1, \ldots, n$, $1 \leq v_i \leq n$, a podmínku

$$v_i - v_j + n x_{ij} \leq n - 1, \quad i = 1, \ldots, n,\ j = 2, \ldots, n. \tag{9.2}$$

[1)] Toto souvisí z tzv. Grahamovým problémem součtu odmocnin. Uvažme dvě řady čísel 1, 25, 31, 84, 87, 134, 158, 182, 198, a 2, 18, 42, 66, 113, 116, 169, 175, 199. Druhá má větší součet odmocnin, ale pokud ke každému číslu přičteme 10^6, tak je součet odmocnin větší jen o $\approx 3 \cdot 10^{-37}$.

(2) *podmínka Dantzig–Fulkerson–Johnson (1954):*

$$\sum_{i\in U}\sum_{j\notin U} x_{ij} \geq 1, \quad \forall U \subseteq \{1,\ldots,n\},\ 1 \leq |U| \leq n-1. \tag{9.3}$$

(3) *podmínka Dantzig–Fulkerson–Johnson (1954) jinak:*

$$\sum_{i\neq j\in U} x_{ij} \leq |U|-1, \quad \forall U \subseteq \{1,\ldots,n\},\ 1 \leq |U| \leq n-1. \tag{9.4}$$

Důkaz.

(1) Sporem, nechť máme bez újmy na obecnosti podcyklus $2,3,\ldots,k$, to jest $x_{23} = x_{34} = \ldots = x_{k-1,k} = x_{k2} = 1$. Pak (9.2) mají instance

$$\begin{aligned} v_2 - v_3 &\leq -1,\\ v_3 - v_4 &\leq -1,\\ &\vdots\\ v_k - v_2 &\leq -1, \end{aligned}$$

a jejich součtem odvodíme $0 \leq -k$, což je spor.

Naopak, nechť nyní x odpovídá hamiltonovské kružnici. Setřiď vrcholy podle pořadí navštívení s tím, že začneme ve vrcholu 1. Přiřaď $v_i := k$ pokud vrchol i je v pořadí k-tý na kružnici (tedy speciálně $v_1 = 1$). Nyní, buďte $i \in \{1,\ldots,n\}$, $j \in \{2,\ldots,n\}$ libovolné. Pokud $x_{ij} = 0$, tak podmínka (9.2) má tvar $v_i - v_j \leq n-1$, což je automaticky splněno. Pokud $x_{ij} = 1$, tak podmínka (9.2) má tvar $v_i - v_j \leq -1$, což je také splněno, protože vrchol j následuje na i na kružnici.

(2) Jasné z toho, že pokud množinu vrcholů rozdělíme na dvě neprázdné části, tak hamiltonovská kružnice musí přejít od jedné do druhé.

(3) Jasné z toho, že uvnitř každé vlastní části grafu je odpovídající část hamiltonovské kružnice pouze stromem (či lesem). □

Pro symetrické TSP pak (9.3) a (9.4) mají tvar

$$\sum_{e\in E(U,V\setminus E)} x_e \geq 2, \quad \forall U \subseteq \{1,\ldots,n\},\ 3 \leq |U| \leq \lfloor n/2 \rfloor,$$

a

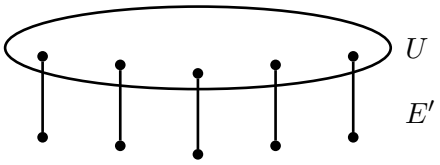
$$\sum_{e\in E(U)} x_e \leq |U|-1, \quad \forall U \subseteq \{1,\ldots,n\},\ 3 \leq |U| \leq \lfloor n/2 \rfloor, \tag{9.5}$$

kde $E(U,U')$ značí množinu hran spojujících vrcholy z U s vrcholy z U', a pro zjednodušení $E(U) := E(U,U)$.

Miller–Tucker–Zemlinova podmínka má výhodu, že omezení není mnoho, ale na druhou stranu nejsou příliš silné (že by byly stěnové ve smyslu pojmu ze strany 9). Naopak, (9.3) resp. (9.4) obsahují exponenciální počet nerovnic, ale jsou zaručeně stěnové pro $n \geq 3$ (Grötschel, Padberg, 1979). Přes vysoký počet nerovnic nemáme zaručeno, že řešení relaxované úlohy je optimální. Přesto se tato formulace (v této nebo jiné formě) používá spíše než Miller–Tucker–Zemlinova.

9.2 Sečné nadroviny

Zaměříme se na podmínku (9.4), která jde následujícím způsobem ještě zesílit [Nemhauser and Wolsey, 1999], a pomůže odříznout další nepřípustná řešení při celočíselné relaxaci. *Hřebenem* nazveme množinu $U \subseteq V$ z (9.4) spolu s množinou hran E', kde hran je lichý počet a každá má právě jeden konec v U. Pak můžeme pro symetrické TSP odvodit následující podmínku. Pokud $|E'| \geq 3$, tak je podmínka stěnová.



Věta 9.7 (Hřebenová nerovnost)**.** *Za výše uvedených předpokladů každé přípustné řešení splňuje*

$$\sum_{e\in E(U,U)} x_e + \sum_{e\in E'} x_e \le |U| + \frac{1}{2}(|E'|-1). \tag{9.6}$$

Důkaz. Uvažujme nerovnosti

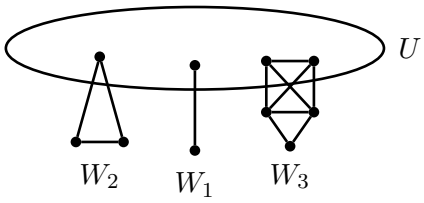
$$\begin{aligned}\sum_{e\colon e=\{i,j\}\in E} x_e &= 2 \quad \forall i \in U,\\ x_e &\le 1 \quad \forall e \in E',\\ -x_e &\le 0 \quad \forall e \in E(U, V\setminus U)\setminus E'.\end{aligned}$$

Součtem dostaneme

$$\sum_{e\in E(U)} 2x_e + \sum_{e\in E'} 2x_e \le 2|U| + |E'|.$$

Vydělením 2 a zaokrouhlením pravé strany dolů pak máme (9.6). □

Podmínku můžeme zobecnit tím, že místo hřebenu budeme uvažovat *zobecněný hřeben* (Chvátal, 1973). V něm zuby hřebenu nahradíme jakýmikoliv podgrafy $W_1,\dots,W_k$ lichého počtu tak, aby byly disjunktní a měli aspoň jeden vrchol uvnitř i vně U. Pak můžeme odvodit následující podmínku.



Věta 9.8 (Zobecněná hřebenová nerovnost)**.** *Za výše uvedených předpokladů každé přípustné řešení splňuje*

$$\sum_{e\in E(U)} x_e + \sum_{i=1}^{k} \sum_{e\in E(W_i)} x_e \le |U| + \sum_{i=1}^{k} |W_i| - \frac{1}{2}(3k+1). \tag{9.7}$$

Důkaz. Jednoduše odvodíme

$$\sum_{e\in E(U)} 2x_e + \sum_{i=1}^{k} \sum_{e\in E(U,V\setminus U)\cap E(W_i)} x_e \le 2|U|,$$

jakožto další nerovnosti typu (9.5):

$$\begin{aligned}\sum_{e\in E(W_i)} x_e &\le |W_i| - 1, \quad \forall i = 1,\dots,k\\ \sum_{e\in E(U\cap W_i)} x_e &\le |U\cap W_i| - 1, \quad \forall i = 1,\dots,k\\ \sum_{e\in E(W_i\setminus U)} x_e &\le |W_i\setminus U| - 1, \quad \forall i = 1,\dots,k.\end{aligned}$$

Součtem dostaneme

$$\sum_{e\in E(U)} 2x_e + \sum_{i=1}^{k} \sum_{e\in E(W_i)} 2x_e \le 2|U| + \sum_{i=1}^{k}(|W_i| - 1 + |U\cap W_i| - 1 + |W_i \setminus U| - 1).$$

Protože $|W_i \setminus U| = |W_i| - |U \cap W_i|$, pravá strana nerovnosti se zjednoduší na

$$2|U| + \sum_{i=1}^{k}(2|W_i| - 3).$$

Vydělením 2 a zaokrouhlením pravé strany dolů pak máme (9.7). □

Protože pravá strana (9.7) se dá vyjádřit jako $|U| + \frac{1}{2}\big(\sum_{i=1}^{k}(2|W_i| - 3) - 1\big)$, je jasné, že se jedná o zobecnění (9.6).

Aby toho nebylo málo, můžeme zobecněné hřebeny ještě dále zobecňovat na tzv. *klikové stromy* (Grötschel–Pulleyblank, 1986). Např. tak, že zuby hřebenu budou vlastní hřebeny, nebo že se různě hřebeny navzájem propojí skrze společné zuby. Platí, že takováto zobecnění generují vesměs stěnové nerovnosti, přesto se tímto postupem k celočíselnému polyedru M_I nemusíme vůbec dopracovat.

Není znám žádný polynomiální algoritmus na generování nesplněných hřebenových nerovností. Na druhou stranu, není zatím ani známo, že by to bylo NP-těžké, takže problém je otevřen.

9.3 Relaxace a dolní odhady na optimální hodnotu

Eukleidovské TSP – kontrolní zóny

Pro eukleidovské TSP můžeme dolní odhad na optimum získat jako optimální hodnotu lineárního programu

$$\max \sum_{i=1}^{n} 2r_i \quad \text{za podm.} \quad r_i + r_j \le c_{ij}\ \forall i,j, \quad r_i \ge 0\ \forall i. \tag{9.8}$$

Libovolné přípustné řešení nám vytvoří kolem každého vrcholu i kružnici s poloměrem r_i. Protože se tyto kružnice neprotínají, k navštívení vrcholu i musím projít kružnicí a urazit vzdálenost aspoň $2r_i$.

Metrické TSP

Minimální kostra grafu určitě dá dolní odhad na optimum. Můžeme ke kostře ještě přidat nejkratší (v kostře neobsaženou) hranu vycházející z libovolného listu kostry. A samozřejmě vybereme list, z něhož vycházející nejkratší hrana (mimo kostru) je co nejdelší

Jiná možnost je uvažovat tzv. minimální 1-strom. Najdeme minimální kostru na grafu bez vrcholu $v = 1$ (nebo jakéhokoli jiného) a přidáme k ní dvě nejkratší hrany vycházející z v. Hodnota takovéhoto minimálního 1-stromu typicky bývá $\approx 10\%$ pod optimem.

Přiřazovací problém

Pokud ve formulaci TSP relaxujeme podmínku (9.1c), dostaneme dopravní problém, resp. jeho speciální případ nazývaný přiřazovací problém. Dopravní problém je řešitelný pomocí lineárního programování nebo speciálních metod (např. tzv. maďarská metoda (König & Egerváry, 1931, Kuhn, 1955), která hledá největší perfektní párování v odpovídajícím bipartitním grafu iterativním zvětšováním menšího párování a s časem $\mathcal{O}(n^4)$ dává silně polynomiální algoritmus).

To, že se jedná o dopravní problém, nahlédneme snadno. Množinu vrcholů zdvojíme $V \cup V'$ a hrany povedou jen z V do V'. Pokud nastavíme kapacity výrobců i spotřebitelů na 1, a ceny jako c_{ij}, dostáváme přímo dopravní problém, viz příklad 1.11.

Přiřazovací problém pro symetrické TSP

Pro symetrické TSP, relaxovaná úloha (i bez podmínek $x_e \leq 1$) má tvar jednoduchého lineárního programu

$$\min \sum_{e \in E} c_e x_e \quad \text{za podm.} \quad \sum_{e \ni k} x_e = 2 \ \forall k \in V, \ x_e \geq 0.$$

Pro něj vždy existuje optimální řešení se složkami v $\{0, 1, 2\}$. Zajímavé jest, že duální úlohou jsou kontrolní zóny (9.8); podmínky $r_i \geq 0 \ \forall i$ tak sice nevzniknou explicitně, ale platí implicitně díky trojúhelníkové nerovnosti.

Pokud přidáme omezení $x_e \leq 1$, $e \in E$, dolní odhad se nám zlepší a optimální řešení bude mít složky v $\{0, \frac{1}{2}, 1\}$.

Celočíselná relaxace podle podcest

Pokud relaxujeme pouze podmínku celočíselnosti, dostaneme také lineární program. Při podmínce (9.3) či (9.4) ale má úloha exponenciální počet nerovnic. Principiálně můžeme i takovouto úlohu řešit polynomiálně, protože dovedeme polynomiálně rozhodnout separační problém. *Separační problém* zní:

> Dán konvexní polyedr M a bod $y \in \mathbb{Q}^n$. Rozhodni, zda $y \in \mathbb{Q}$, a pokud ne, tak najdi oddělující nadrovinu, tj. $a \in \mathbb{Q}^n$ takové, že $a^T x < a^T y \ \forall x \in M$.

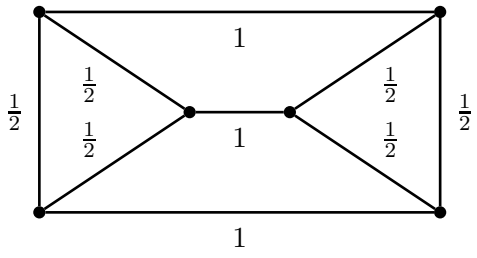
Pokud je separační problém řešitelný efektivně, pak i jakýkoli lineární program nad M, a to s využitím elipsoidového algoritmu [Schrijver, 1998].

Uvažme např. podmínku (9.3), pokaženou podmínku pro separační problém najdeme pomocí efektivních algoritmů pro toky v sítích. Uvažme náš graf jako síť s kapacitami x_{ij} a najdeme maximální tok mezi vrcholy 1 a $j \in V$. Maximální tok nám dává zároveň i minimální řez. Je-li jeho hodnota alespoň 1, tak všechny podmínky z (9.3) jsou splněny, jinak minimální řez dává rovnou množinu U. Nestačí však řešit pouze jednu úlohu, ale musíme (pokud ještě nemáme porušenou podmínku) projít všechny toky z vrcholu 1 do jakéhokoli jiného, a pro nesymetrické TSP i v opačném směru.

Aproximační faktor je 1.5, to jest, optimální cesta není delší než 1.5-násobek dolní meze.

Následující příklad ukazuje, že i přes exponenciální počet podmínek celočíselná relaxace nemusí dávat celočíselné řešení.

Příklad 9.9. Uvažujme následující graf. Čísla na hranách udávají hodnotu přípustného řešení x. Hrany, které v grafu nejsou znázorněny, mají příslušnou hodnotu $x_e = 0$.



Příslušné řešení x je přípustné řešení celočíselné relaxace, tudíž splňuje všechna omezení typu (9.3) či (9.4). Nicméně, toto řešení stále ještě není celočíselné. □

Lagrangeova relaxace

Potenciálně lepší horní odhad než je celočíselná relaxace nám může dát Lagrangeova relaxace (sekce 5.2). Uvažujme symetrické TSP a analogii formulace (9.4), navíc s redundantní podmínkou $\sum_e x_e = n$.

$$\begin{aligned}\min \sum_{e\in E} c_e x_e \quad \text{za podm.} \quad & \sum_{e\ni i} x_e = 2 \ \forall i \in \{1,\dots,n\},\\ & \sum_{e\in E(U)} x_e \le |U|-1, \quad \forall 1 \notin U \subset \{1,\dots,n\},\ 2 \le |U| \le n-1,\\ & \sum_{e\in E} x_e = n,\\ & x \in \{0,1\}^{|E|}.\end{aligned}$$

Lagrangeovsky relaxujeme rovnice pro všechna $i \neq 1$ a dostaneme úlohu (P_u)

$$\begin{aligned}\min \sum_{e\in E} c_e x_e + \sum_{i=2}^{n} 2u_i - & \sum_{1\notin e=\{i,j\}\in E} (u_i+u_j)x_e\\ \text{za podm.} \quad & \sum_{e\ni 1} x_e = 2,\\ & \sum_{e\in E(U)} x_e \le |U|-1, \quad \forall 1 \notin U \subset \{1,\dots,n\},\ 2 \le |U| \le n-1,\\ & \sum_{e\in E} x_e = n,\\ & x \in \{0,1\}^{|E|}.\end{aligned}$$

Přípustná řešení této úlohy jsou právě 1-stromy. Minimální 1-strom najdeme jednoduše jako minimální kostru na podgrafu na $\{2,\dots,n\}$ a přidáme dvě nejkratší hrany z vrcholu 1. Tedy úlohu (P_u) dovedeme efektivně řešit, což je v Lagrangeově relaxaci to podstatné.

Podle Důsledku 5.7 dostaneme stejný odhad jako celočíselnou relaxací. Nemusíme se ale vypořádávat s exponenciálním počtem podmínek. Proto je tato relaxace je asi nejúspěšnější pro dolní odhady TSP. Původně pochází již od autorů Held–Karp (1970, 1971).

9.4 Heuristiky

Heuristiky dávají více či méně dobré přípustné řešení a tím pádem i horní odhad na optimální hodnotu TSP. Předpokládáme zde metrické TSP. Známé heuristiky jsou např.

Nejbližší soused

Vyber počáteční vrchol. Přesun se do nejbližšího sousedního vrcholu, z něj pak do jemu nejbližšímu sousedu (kterého jsme ještě nenavštívili) atd. dokud nepokryjeme všechny vrcholy a cestu neuzavřeme.

Časová složitost je $\mathcal{O}(n^2)$, aproximační faktor $\frac{1}{2}(\lceil \log(n) \rceil + 1)$ a nelze zlepšit.

Hladový algoritmus

Setřiď hrany vzestupně dle délek a postupně buduj kružnici přidáváním nejkratších hran tak, abychom neporušili přípustnost, to jest, aby nevznikla podkružnice či vrchol stupně 3.

Časová složitost je $\mathcal{O}(n^2 \log(n))$. Pro větší n se chová lépe než Nejbližší soused, aproximační faktor $\frac{1}{2}(\lceil \log(n) \rceil + 1)$.

Vkládací algoritmus

Induktivně přidávám vrcholy do již zbudované kružnice. Vkládám tak, aby podcesta vzrostla co nejméně. Přidávaný vrchol může být: to s minimálním nárůstem cesty či to nejbližší nějakému

vrcholu na kružnici (aproximační faktor 2), anebo to s největší vzdáleností od vrcholů na kružnici (aproximační faktor $\sim \log(n)$, ale chová se dobře a robustně) [Cook, 2012].

Popíšeme podrobněji variantu přidání nejbližšího vrcholu k nějakému vrcholu na kružnici T (Nearest insertion, Nemhauser and Wolsey [1999]). Definuj $(i^*, j^*) \coloneqq \arg\min_{i \in V \setminus T,\ j \in T} c_{ij}$. Pak ke kružnici T přidáme hranu $\{i^*, j^*\}$ a vrchol i^* napojíme na toho výhodnějšího ze dvou sousedů vrcholu j^*.

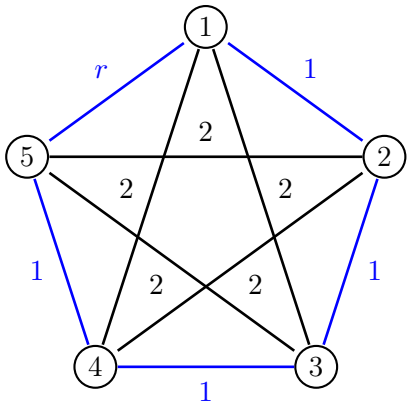
Christofides (1976)

Christofidův algoritmus najde řešení s aproximačním faktorem 3/2. Nejprve si ukažme metodu s faktorem 2: Najdi minimální kostru a zdvojnásob její hrany, takže na nich existuje eulerovský tah (jehož délka je nanejvýš optimum TSP). Tím, že přeskočíme/vyškrtneme vícenásobná navštívení téhož vrcholu celkovou délku nezvýšíme (díky trojúhelníkové nerovnosti) a dostaneme hamiltonovskou kružnici.

Vylepšení na faktor 3/2 provedeme tak, že k nalezené minimální kostře přidáme minimální párování na vrcholech kostry lichého stupně, což jde najít polynomiálně. Faktor 3/2 pak plyne z toho, že minimální párování má cenu maximálně poloviční jak TSP (každá hamiltonovská kružnice se rozpadá na dvě disjunktní párování).

Heuristik existuje ještě celá řada. Nejmenovali jsme třeba ještě metody: Savings (Clarke–Wright, 1964), Sweep pro eukleidovské TSP či Borůvkův algoritmus.

Příklad 9.10 (bez trojúhelníkové nerovnosti)**.** Uvažme graf s $n = 5$, kde hrany na cestě 1-2-3-4-5 jsou ohodnoceny 1, hrana $(5, 1)$ má délku r a ostatní 2.



Metoda nejbližšího souseda iniciovaná ve vrcholu 1 najde kružnici 1-2-3-4-5-1 délky $4 + r$ (na obrázku vyznačeno modře), ale optimální je 7. Tedy pro dost velké r dostaneme libovolně špatný aproximační faktor. To je důvod proč pro heuristiky většinou předpokládáme metrické TSP. □

Kromě heuristik na nalezení dobrého přípustného řešení existují i lokálně vylepšující algoritmy, které začnou s počátečním přípustným řešením a iterativně se ho pokoušejí zlepšit lokálními změnami.

k-výměna

Kromě heuristik na nalezení výchozího přípustného řešení existují také lokálně vylepšující techniky. Např. k-výměna spočívá v odstranění k hran z kružnice a přidání nových k hran tak, aby vznikla zase hamiltonovská kružnice. Pokud je nová kružnice lepší, zapamatujeme si ji. Vyzkoušíme-li všechny takovéto k-výměny, dostaneme jako řešení „lokální minimum“. Počet k-výměn může být ale velký, takže se časově vyplatí jen pro malá k. Tabulka ukazuje počet možných napojení po odstranění k hran

k	2	3	4	5	6	7	8
počet přeuspořádání	1	4	20	148	1358	15104	198144

Lin–Kernighan–Helsgaun (LKH)

Namísto zkoušení všech možných k-výměn přišli Lin–Kernighan (1973) s variabilní k-výměnou, založenou na sekvenční výměně. Efektivní implementaci pak navrhnul Helsgaun [2000].

Sekvenční výměnu si lze představit takto. Z hamiltonovské kružnice odstraníme hranu $x_1 = (t_1, t_2)$ a dostaneme hamiltonovskou cestu. Když nyní přidáme hranu $y_1 = (t_2, t_3)$, je jen jediná volba jak odstranit hranu $x_2 = (t_3, t_4)$, abychom zase dostali cestu. Tento postup „přidání hrany a odstranění hrany“ opakujeme tak dlouho než uzavření cesty na kružnici nezlepší dosavadní nejlepší řešení nebo bychom se opakovali. V druhém případě backtrackujeme zpět a zkoušíme jiné volby hran.

Zefektivnit tento postup lze mnoha způsoby. Následující lemma říká, že stačí uvažovat pouze takové výběry hran y_k, aby částečné součty vah hran $\sum_{i=1}^{k} g_i > 0$, kde $g_i \coloneqq c(x_i) - c(y_i)$.

Lemma 9.11. *Buďte $g_0, \ldots, g_{p-1} \in \mathbb{R}$ tak, že $\sum_{i=0}^{p-1} g_i > 0$. Pak existuje i^* pro nějž částečné součty $\sum_{i=0}^{k} g_{i^*+i} > 0 \ \forall k = 0, \ldots, p-1$, kde indexy sčítáme modulo p.*

Důkaz. Definujme

$$\begin{aligned} r_1 &\coloneqq \min\{r \geq 0;\ \textstyle\sum_{i=0}^{r} g_i \leq 0\},\\ r_2 &\coloneqq \min\{r \geq r_1 + 1;\ \textstyle\sum_{i=r_1+1}^{r} g_i \leq 0\},\\ &\vdots\\ r_\ell &\coloneqq \min\{r \geq r_{\ell-1} + 1;\ \textstyle\sum_{i=r_{\ell-1}+1}^{r} g_i \leq 0\}. \end{aligned}$$

Nyní stačí volit $i^* \coloneqq r_\ell + 1$, protože podle definice posledního r_ℓ je $\sum_{i=i^*}^{k} g_i > 0$ pro všechna $k = i^*, \ldots, p-1$. Podle předpokladu je $\sum_{i=0}^{p-1} g_i > 0$, tudíž $\sum_{i=i^*}^{p-1} g_i + \sum_{i=0}^{k} g_i > 0$ pro všechna $k = 0, \ldots, i^* - 1$. □

Dále, výběr y_i je omezen na pět hran nejbližších minimálnímu 1-stromu. Kromě sekvenčních výměn také zkouší nesekvenční 4- a 5-výměny (tvar květu). Takováto 4-výměna se dá použít i jako popostrčení do nové vylepšující sekvence místo zkoušení nových začátků s náhodnými cestami (tzv. zřetězený algoritmus).

Uvádí se, že pro $n < 50$ najde LKH optimum skoro ve všech případech a pro $n \approx 100$ v ca 30% případů. A i pokud nenajde optimum, tak nebývá víc než 1% až 2% od optimální hodnoty. V současnosti (2011) drží rekord o nejlepší řešení World TSP, úlohy s 85 900 městy za Zemi.

Další

Simulované žíhání bylo původně vymyšleno právě pro TSP a NP-těžké problémy obecně [Kirkpatrick et al., 1983].

Genetické algoritmy. Zde je otázka jak křížit jedince (=cesty). Metoda EAX (Edge-Assembly Crossover) vezme sjednocení cest, a u kružnic se střídavými hranami vezme hrany 2. cesty a nahradí jimi ty doplňkové hrany u celé 1. cesty. Varianta Nagata [2006] našla zatím nejlepší řešení u 100 000 vrcholového grafu Mony Lisy.

9.5 Historie a výpočty

Na rozdíl od problému batohu je TSP je skutečně stále těžké vyřešit k optimu i pro středně velké instance. Tabulka dole ilustruje vývoj při řešení TSP na největších vyřešených instancích své doby. Čas od roku 2000 je souhrnný čas všech výpočetních jednotek při paralelním běhu.

rok	1957	1987	1998	2001	2004	2006
n	49	666	13509	15112	24978	85900
popis	státy USA	města světa	města USA	města Německa	města Švédska	LaserLogic chip
čas				23 let	84 let	

Vítězem (nejen) instance z roku 2006 je program Concorde, viz Cook [2011]. Program je založený na metodě branch & cut, a používají se dobré heuristiky typu Lin–Kernighan–Helsgaun.

Úloha TSP o nalezení cesty 1 904 711 městy Země z r. 2001 má Helsgaunem nalezené řešení vzdálené max. 0.047% od optima. Jedna z největších instancí TSP z r. 2003 se skládá z 526 280 881 nebeských objektů (Cook & Applegate). Zatím nejlepší nalezená cesta mezi hvězdami je nanejvýš 0.41% od optima.

Z historie

Na přelomu 19. a 20. století projíždělo Spojenými státy na 350 000 obchodních cestujících. Jeden z nich, H.M. Cleveland roku 1925 procestoval od 9.7. do 24.8. celkem 350 míst v Maine [Cook, 2012].

Plán okružní cesty zajímal i právníky navštěvující soudní okrsky ve svých obvodech. Jedním z nich byl i budoucí prezident A. Lincoln, který kolem roku 1850 spravoval 14 místních soudních dvorů v Osmém obvodě ve státě Illinois.

Podobné okružní cesty vykonávali (a svým způsobem někteří ještě vykonávají) i kazatelé.

Označení TSP pro tento problém se poprvé objevilo v tisku roku 1949 [Robinson, 1949].

Cvičení

9.1. Dokažte, že dimenze celočíselného polyedru (konvexní obal celočíselných přípustných bodů) z úlohy (9.1) je rovna $\frac{1}{2}n(n-3)$, a to například tak, že najdete $\frac{1}{2}n(n-3)+1$ afinně nezávislých přípustných řešení.

9.2. Navrhněte jak polynomiálně řešit separační problém v celočíselné relaxaci i pro exponenciální formulaci (9.4).

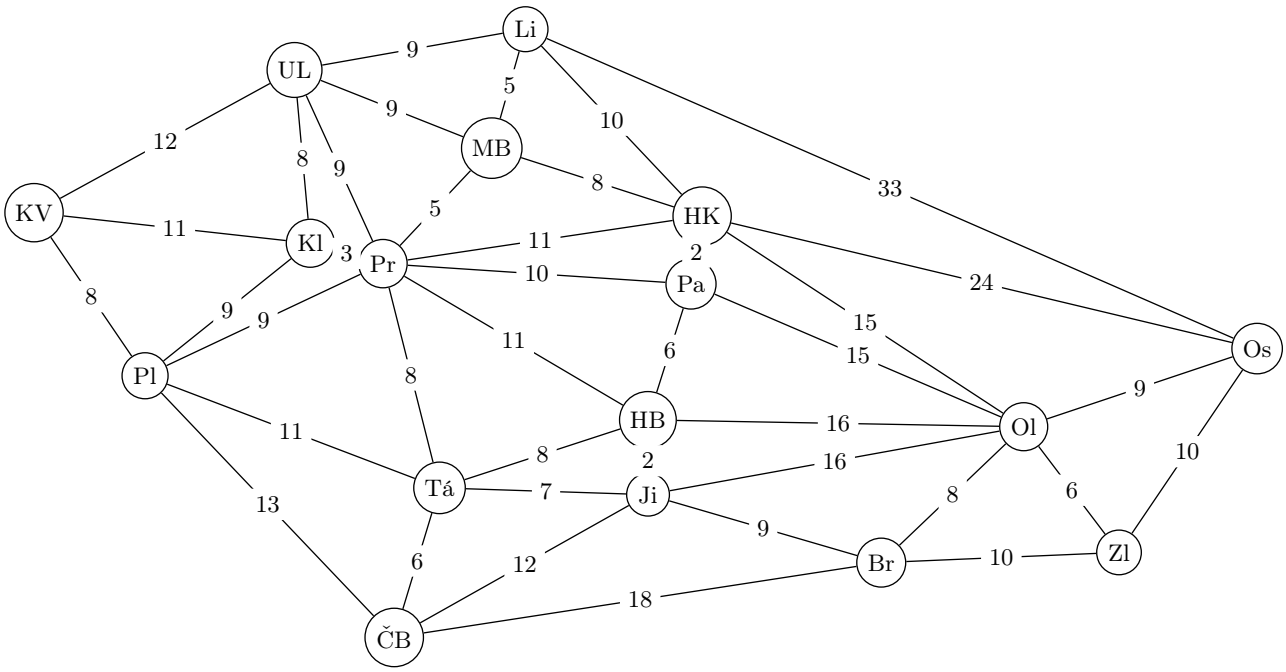
9.3. Pro symetrické TSP s $n \geq 3$ dokažte, že nerovnice (9.5) je stěnová, a to například tak, že najdete $\frac{1}{2}n(n-3)-1$ afinně nezávislých přípustných řešení.

9.4. Pro úlohu s $n = 4$ najděte všechny stěnové nerovnosti.

9.5. Aplikujte Lagrangeovu relaxaci na úlohu TSP s tím, že relaxujete jiné podmínky.

Zejména, co se stane, když relaxujeme *všechny* rovnicové podmínky?

9.6. Najděte co nejkratší okružní cestu po městech České republiky (vzdálenosti mají za jednotku 10 km). Aplikujte různé heuristiky i přesný výpočet.



Kapitola 10

Facility location problem

Facility location problem (FLP), nebo též plant location problem a do češtiny občas překládaný jako problém umístění zařízení, se zabývá otázkou kolik a jaké ze zařízení $i = 1, \ldots, m$ vybudovat, abychom uspokojili spotřebitele $j = 1, \ldots, n$ a celkem nás to stálo co nejméně peněz. Označme: M_i kapacita i-tého zařízení a f_i cena jeho vybudování, c_{ij} cena za převoz jednotky zboží od i ku j, a d_j požadavek j-tého spotřebitele. Proměnná x_{ij} bude udávat množství zboží přepravované od i ku j, a binární proměnná y_i udává zda se i-té zařízení postaví či ne. Nyní úlohu můžeme formulovat jako celočíselný lineární program

$$\min \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{i=1}^{m} f_i y_i \quad \text{za podm.} \quad \sum_{i=1}^{m} x_{ij} = d_j \quad \forall j, \tag{10.1a}$$

$$\sum_{j=1}^{n} x_{ij} \le M_i \quad \forall i, \tag{10.1b}$$

$$x_{ij} \le M_i y_i \quad \forall i, j, \tag{10.1c}$$

$$x_{ij} \ge 0 \quad \forall i, j, \tag{10.1d}$$

$$y \in \{0,1\}^m. \tag{10.1e}$$

Podmínku (10.1b)–(10.1c) můžeme ekvivalentně nahradit jedním typem nerovností

$$\sum_{j=1}^{n} x_{ij} \le M_i y_i \quad \forall i,$$

nicméně není tak výhodná. Přestože je počet omezení ve formulaci (10.1) větší, jsou silnější, a proto je formulace (10.1b)–(10.1c) výhodnější.

Kromě tohoto modelu budeme ještě uvažovat případ bez kapacit (UFLP, *uncapacitated facility location problem*). Zde stačí podmínky (10.1b)–(10.1c) nahradit podmínkou $x_{ij} \le (\sum_{j=1}^{n} d_j) y_i$. Elegantnější je ale zavést si nové proměnné $z_{ij} \coloneqq x_{ij}/d_j$ udávající jaký podíl požadavku j-tého spotřebitele pochází od i. Substitucí $c'_{ij} \coloneqq d_j c_{ij}$ pak dostáváme model

$$\min \sum_{i=1}^{m}\sum_{j=1}^{n} c'_{ij}z_{ij} + \sum_{i=1}^{m} f_i y_i \quad \text{za podm.} \quad \sum_{i=1}^{m} z_{ij} = 1 \quad \forall j, \tag{10.2a}$$

$$z_{ij} \le y_i \quad \forall i, j, \tag{10.2b}$$

$$z_{ij} \ge 0 \quad \forall i, j, \tag{10.2c}$$

$$y \in \{0,1\}^m. \tag{10.2d}$$

Zde opět bychom mn podmínek v (10.2b) mohli nahradit m podmínkami

$$\sum_{j=1}^{n} z_{ij} \le n y_i \quad \forall i, \tag{10.3}$$

nicméně ty původní jsou stěnové, a tudíž i mnohem vhodnější pro řadu metod. Jeden knižní zdroj uváděl numerické experimenty, kdy model s podmínkou (10.3) trval 15 hodin výpočtu, kdežto s podmínkou (10.2b) pouze 2 sekundy. Tudíž na modelování velmi záleží!

Poznámka 10.1. Buď $y \in \{0,1\}^m$ optimální řešení v úloze UFLP. Pak optimální rozvozy z_{ij} spočítáme jednoduše hladovým algoritmem. Buď $j \in \{1,\dots,n\}$ libovolné a vyber $i^* \in \arg\min_{i:\, y_i=1} c'_{ij}$. Pak optimální řešení je $z_{i^*j} = 1$ a $z_{ij} = 0$ pro $i \neq i^*$. Tudíž existuje optimální řešení UFLP takové, že všechny složky y_i, z_{ij} jsou binární.

Příklad 10.2. Chceme ukázat, že podmínky (10.2b) jsou stěnové pro speciální případ s $m = 1$. Přesněji řečeno, aby úloha nebyla tak degenerovaná, uvažujeme trochu jiné omezení typu

$$e^T z \leq y, \quad z \leq e, \quad z \in \mathbb{R}^n, \ y \in \{0,1\},$$

a dokážeme, že podmínka $z_i \leq y$ je stěnová pro každé $i = 1,\dots,n$.

Řešení: Stačí vzít $n+1$ bodů $(0,0)$, $(e_i,1)$ a $(e_i+e_j,1)$ pro všechna $j \neq i$. Body zřejmě splňují zadaná omezení a leží v nadrovině $z_i = y$.

Nyní musíme ověřit, že jsou afinně nezávislé. Připomeňme, že body $x_0, x_1, \dots, x_n$ jsou afinně nezávislé právě tehdy, když body $x_1 - x_0, \dots, x_n - x_0$ jsou lineárně nezávislé. Volbou $x + 0 := (0,0)$ se problém redukuje na lineární nezávislost bodů $(e_i,1)$ a $(e_i+e_j,1)$, $j \neq i$, což se nahlédne snadno. □

10.1 Relaxace

Celočíselnou relaxací dostaneme jako vždy lineární program. Nicméně pro úlohu UFLP s podmínkami (10.3) můžeme optimum celočíselné relaxace snadno vyčíst. V tomto případě má úloha tvar

$$\begin{aligned} \min \sum_{i=1}^m \sum_{j=1}^n c'_{ij} z_{ij} + \sum_{i=1}^m f_i y_i \ \text{ za podm. } & \sum_{i=1}^m z_{ij} = 1 \quad \forall j, \\ & \sum_{j=1}^n z_{ij} \leq n y_i \quad \forall i, \\ & z_{ij} \geq 0 \quad \forall i,j, \\ & y_i \geq 0 \quad \forall i. \end{aligned}$$

Protože koeficienty f_i v účelové funkci jsou z podstaty problému kladné, chceme proměnné y_i co nejmenší. Tudíž se nerovnost $\sum_{j=1}^n z_{ij} \leq n y_i$ nabyde jako rovnost. Substitucí $y_i := \frac{1}{n}\sum_{j=1}^n z_{ij}$ dostaneme úlohu

$$\min \sum_{i=1}^m \sum_{j=1}^n \left(c'_{ij} + \frac{1}{n} f_i \right) z_{ij} \ \text{ za podm. } \ \sum_{i=1}^m z_{ij} = 1 \ \forall j, \quad z_{ij} \geq 0 \ \forall i,j.$$

Tato úloha je ekvivalentně vyjádří

$$\sum_{j=1}^n \min \sum_{i=1}^m \left(c'_{ij} + \frac{1}{n} f_i \right) z_{ij} \ \text{ za podm. } \ \sum_{i=1}^m z_{ij} = 1, \quad z_{ij} \geq 0 \ \forall i.$$

Úloha je tedy separabilní, rozdělí se na n podúloh. Pro konkrétní $j \in \{1,\dots,n\}$ má j-tá podúloha tvar

$$\min \sum_{i=1}^m \left(c'_{ij} + \frac{1}{n} f_i \right) z_{ij} \ \text{ za podm. } \ \sum_{i=1}^m z_{ij} = 1, \quad z_{ij} \geq 0 \ \forall i.$$

Buď $i^* \in \arg\min_{i=1,\dots,m} \left(c'_{ij} + \frac{1}{n} f_i \right)$. Pak optimální řešení podúlohy, a tudíž i celočíselné relaxace, je $z_{i^*j} = 1$ a $z_{ij} = 0$ pro $i \neq i^*$.

10.2 Branch & bound

Jisté specifikum metody Branch & bound v úloze FLP (10.1) může být volba proměnné $y_k \notin \mathbb{Z}$ pro větvení. Rozumné volby jsou:

- $k \coloneqq \arg\max_i M_i$, protože rychle najdeme přípustné řešení.
- $k \coloneqq \arg\min_i \frac{f_i}{M_i}$, protože navíc zahrneme i cenu výstavby i-tého zařízení.
- $k \coloneqq \arg\min_i \frac{f_i + \sum_{j=1}^{n} c_{ij} d_j}{M_i}$, protože navíc zahrneme i ceny za převoz zboží.

10.3 Heuristiky pro UFLP

Aneb jak najít rychle co nejlepší přípustné řešení.

Lineární relaxace

Relaxací celočíselných podmínek $y \in \{0,1\}^m$ na spojité $y' \in [0,1]^m$ se úloha redukuje na lineární program, jehož optimální hodnota dá horní odhad na původní optimální hodnotu, a z optimálního řešení y' můžeme dostat přípustné řešení původní úlohy prostým zaokrouhlením nahoru $y \coloneqq \lceil y' \rceil$. Protože ale typicky $m \ll n$, tak bude přípustné řešení $y = e$, což většinou je silně nadhodnocené.

Hladový algoritmus

Vytvářet přípustné $y \in \{0,1\}^m$ můžeme také hladově. Na začátku položíme $y = 0$, a pak postupně přidáváme taková zařízení, která co nejvíce zmenší celkovou cenu.

Časová složitost je $\mathcal{O}(m^2 n)$, pokud si pro dané y pamatujeme i příslušné řešení z.

Lokální vylepšování

Začneme s libovolným přípustným $y \in \{0,1\}^m$ a testujeme zda nedostaneme lepší řešení změnou jednoho bitu y, tj. přidáním či odebráním jednoho zařízení.

Příklad 10.3. Uvažujme úlohu UFLP s daty

$$C' = \begin{pmatrix} 3 & 9 & 2 & 6 \\ 5 & 9 & 7 & 6 \\ 0 & 7 & 6 & 6 \\ 6 & 7 & 4 & 0 \end{pmatrix}, \quad f = \begin{pmatrix} 3 \\ 2 \\ 3 \\ 3 \end{pmatrix}.$$

Najděte optimální řešení, optimální řešení obou relaxací, a aplikujte různé heuristiky.

Řešení: Optimální řešení je

$$y^* = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}, \quad Z^* = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix}.$$

a optimální cena je 17.

Relaxace s podmínkou tvaru (10.2b) vede přímo na optimum, zatímco relaxace s podmínkou (10.3) vede na

$$y^R = \begin{pmatrix} \frac{1}{4} \\ 0 \\ \frac{1}{2} \\ \frac{1}{4} \end{pmatrix}, \quad Z^R = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

s cenou 12.

Zaokrouhlením relaxovaného optima dostaneme přípustné řešení $y^1 = \lceil y^R \rceil = (1,0,1,1)^T$, kterému odpovídá hodnota účelové funkce 18.

Hladový algoritmus pracuje takto: V první iteraci spočítáme ceny přípustných řešení:

$$\begin{aligned} y &= (1,0,0,0)^T : \text{ cena je } 23, \\ y &= (0,1,0,0)^T : \text{ cena je } 29, \\ y &= (0,0,1,0)^T : \text{ cena je } 22, \\ y &= (0,0,0,1)^T : \text{ cena je } 20. \end{aligned}$$

Vybereme tedy poslední, nejlevnější možnost. V druhé iteraci můžeme zařízení přidat na pozice $i \in \{1,2,3\}$:

$$\begin{aligned} y &= (1,0,0,1)^T : \text{ cena je } 19, \\ y &= (0,1,0,1)^T : \text{ cena je } 21, \\ y &= (0,0,1,1)^T : \text{ cena je } 17. \end{aligned}$$

Vybereme opět poslední řešení a nyní můžeme zařízení přidat pouze na pozice $i \in \{1,2\}$:

$$\begin{aligned} y &= (1,0,1,1)^T : \text{ cena je } 19, \\ y &= (0,1,1,1)^T : \text{ cena je } 18. \end{aligned}$$

Protože se ceny zhoršily, zůstaneme u řešení $y = (0,0,1,1)^T$ s cenou 17, které je shodou okolností optimální.

Pro lokální vylepšování začneme s výše uvedeným řešením $y^1 = (1,0,1,1)^T$. Změnou jedné složky $0 \leftrightarrow 1$ dostaneme možná řešení

$$\begin{aligned} y &= (0,0,1,1)^T : \text{ cena je } 17, \\ y &= (1,1,1,1)^T : \text{ cena je } 20, \\ y &= (1,0,0,1)^T : \text{ cena je } 18, \\ y &= (1,0,1,0)^T : \text{ cena je } 21. \end{aligned}$$

Nejlepší je tedy řešení $y = (0,0,1,1)^T$ s cenou 17. V druhé iteraci prozkoumáme

$$\begin{aligned} y &= (0,1,1,1)^T : \text{ cena je } 19, \\ y &= (0,0,0,1)^T : \text{ cena je } 20, \\ y &= (0,0,1,0)^T : \text{ cena je } 22. \end{aligned}$$

Žádnou změnou složky už se tedy cena nezlepší. □

10.4 Bendersova dekompozice pro UFLP

Aplikujme postup popsaný v Sekci 5.1 na model (10.2). Úlohu (10.2) přepíšeme jako

$$\min_{y \in \{0,1\}^m} \left\{ \sum_{i=1}^{m} f_i y_i + \min \left\{ \sum_{i=1}^{m} \sum_{j=1}^{n} c'_{ij} z_{ij}; \sum_{i=1}^{m} z_{ij} = 1 \ \forall j,\ z_{ij} \le y_i \ \forall i,j,\ z_{ij} \ge 0 \ \forall i,j \right\} \right\}.$$

Protože je vnitřní lineární program separabilní, rozpadne se na součet n jednotlivých lineárních programů

$$\min_{y \in \{0,1\}^m} \left\{ \sum_{i=1}^{m} f_i y_i + \sum_{j=1}^{n} P_j(y) \right\}$$

kde $P_j(y)$ má tvar

$$\min \sum_{i=1}^{m} c'_{ij} z_{ij} \text{ za podm. } \sum_{i=1}^{m} z_{ij} = 1,\ z_{ij} \le y_i\ \forall i,\ z_{ij} \ge 0\ \forall i. \qquad P_j(y)$$

Duální úloha k $P_j(y)$ má tvar

$$\max\ u - \sum_{i=1}^{m} y_i w_i \text{ za podm. } u - w_i \le c'_{ij}\ \forall i,\ \ w_i \ge 0\ \forall i. \qquad D_j(y)$$

Protože $w_i \ge 0$ a také $w_i \ge u - c'_{ij}$, a zároveň má být w_i co nejmenší, pro optimální w_i musí platit $w_i = (u - c'_{ij})^+$. Aby (u, w) byl vrchol duálního polyedru, musí rovněž $u = c'_{kj}$ pro nějaké $k \in \{1, \dots, m\}$ (jinak mohu u posunout nahoru či dolu, a w se změní také lineárně). Tedy optimálními vrcholy mohou být jen vektory

$$(u, w_1, \dots, w_m) = (c'_{kj}, (c'_{kj} - c'_{1j})^+, \dots, (c'_{kj} - c'_{mj})^+), \quad k = 1, \dots, m.$$

Pokud $y \neq 0$, což můžeme směle předpokládat, je primární polyedr neprázdný, a tedy duální úloha $D_j(y)$ omezená. Proto nebudeme neomezené směry uvažovat a Bendersova dekompozice vede na formulaci

$$\min \sum_{i=1}^{m} f_i y_i + \sum_{j=1}^{n} \alpha_j \text{ za podm. } \alpha_j \ge c'_{kj} - \sum_{i=1}^{m} (c'_{kj} - c'_{ij})^+ y_i,\ y \in \{0,1\}^m,\ e^T y \ge 1,\ \alpha \in \mathbb{R}^n.$$

Přestože počet binárních proměnných zůstal stejný, počet spojitých proměnných z_{ij} se zmenšil z mn na pouhých n.

10.5 Lagrangeova relaxace pro UFLP

Lagrangeova relaxace podmínek $\sum_{i=1}^{m} z_{ij} = 1$, $j = 1, \dots, n$, modelu (10.2) vede na úlohu

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} (c'_{ij} - u_j) z_{ij} + \sum_{i=1}^{m} f_i y_i + \sum_{j=1}^{n} u_j \text{ za podm. } z_{ij} \le y_i\ \ \forall i, j,\ z_{ij} \ge 0\ \ \forall i, j,\ y \in \{0,1\}^m.$$

Úloha se dá rozdělit na m nezávislých podúloh a ekvivalentně vyjádřit jako

$$\sum_{i=1}^{m} (P_u^i) + \sum_{j=1}^{n} u_j,$$

kde

$$\min \sum_{i=1}^{m} (c'_{ij} - u_j) z_{ij} + f_i y_i \text{ za podm. } 0 \le z_{ij} \le y_i\ \forall j,\ y_i \in \{0,1\}. \qquad (P_u^i)$$

Optimální řešení (P_u^i) se dá vyjádřit explicitně. Je-li $y_i = 0$, tak $z_{ij} = 0\ \forall j$. Je-li $y_i = 1$, tak pokud $c'_{ij} < u_j$, volíme $z_{ij} = 1$, jinak $z_{ij} = 0$. Optimální hodnota je pak

$$\left(\sum_{i=1}^{m} (c'_{ij} - u_j)^- + f_i \right)^- = \min \left(\sum_{i=1}^{m} \min(c'_{ij} - u_j, 0) + f_i, 0 \right).$$

Tudíž úloha (P_u) z Lagrangeovy relaxace má jednoduchý tvar.

Cvičení

Pro úlohu UFLP:

10.1. Dokažte, že konvexní obal množiny přípustných bodů má dimenzi $mn + m - n$.

10.2. Aplikujte Bendersovu dekompozici pro formulaci úlohy s podmínkou (10.3) namísto (10.2b) a porovnejte oba přístupy.

10.3. Aplikujte Lagrangeovu relaxaci (a porovnejte ji s klasickou celočíselnou relaxací) s tím, že se relaxují podmínky $x_{ij} \leq y_j \ \forall i, j$.

Pro úlohu FLP:

10.4. Aplikujte Bendersovu dekompozici pro určitou formulaci.

10.5. Rozmyslete si heuristiky pro nalezení dobrého přípustného řešení, zejména hladový algoritmus a lokální vylepšování a jejich časovou složitost.

10.6. Diskutujte metodu branch & bound (volby proměnné na větvení apod.).

10.7. Aplikujte Lagrangeovu relaxaci (a porovnejte ji s klasickou celočíselnou relaxací). Které podmínky jsou nejvýhodnější k relaxaci?

Značení

Množiny

$\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$	množina přirozených, celých, racionálních a reálných čísel
$U + V$	součet množin, $U + V = \{u + v;\ u \in U, v \in V\}$
$\operatorname{conv}(M)$	konvexní obal množiny M, $\operatorname{conv}(M) = \left\{ \sum_{i=1}^{k} \alpha_i x_i;\ x_i \in M,\ \alpha_i \geq 0,\ \sum_{i=1}^{k} \alpha_i = 1,\ k \in \mathbb{N} \right\}$
$\operatorname{conv}\operatorname{cone}(M)$	konvexní kuželový obal množiny M, $\operatorname{conv}\operatorname{cone}(M) = \left\{ \sum_{i=1}^{k} \alpha_i x_i;\ x_i \in M,\ \alpha_i \geq 0,\ k \in \mathbb{N} \right\}$

Matice a vektory

$\operatorname{rank}(A)$	hodnost matice A
A^T	transpozice matice A
A_{i*}	i-tý řádek matice A
A_{*j}	j-tý sloupec matice A
$\operatorname{diag}(v)$	diagonální matice s diagonálními prvky $v_1, \ldots, v_n$
0_n, 0	nulová matice (všechny složky jsou rovny 0)
1_n	jedničková matice (všechny složky jsou rovny 1)
I_n, I	jednotková matice (diagonální s jedničkami na diagonále)
e_i	jednotkový vektor, $e_i = I_{*i} = (0, \ldots, 0, 1, \ldots, 0)^T$
e	vektor ze samých jedniček, $e = (1, \ldots, 1)^T$

Čísla

$\lfloor r \rfloor$	(dolní) celá část reálného čísla, $\lfloor r \rfloor = \max\{z \in \mathbb{Z};\ z \leq r\}$
$\lceil r \rceil$	horní celá část reálného čísla, $\lceil r \rceil = \min\{z \in \mathbb{Z};\ z \geq r\}$
$\{r\}$	zlomková část reálného čísla, $\{r\} = r - \lfloor r \rfloor$
r^+	kladná část reálného čísla, $r^+ = \max(r, 0)$
r^-	záporná část reálného čísla, $r^- = \max(-r, 0)$

Literatura

T. Achterberg. Scip: Solving constraint integer programs. *Math. Program. Comput.*, 1(1):1–41, 2009. http://mpc.zib.de/index.php/MPC/article/view/4. 43

T. Achterberg and T. Berthold. Improving the feasibility pump. *Discrete Optim.*, 4(1):77 – 86, 2007. http://dx.doi.org/10.1016/j.disopt.2006.10.004. 47

D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook. *The traveling salesman problem. A computational study.* Princeton University Press, 2006. 71

S. Arora. Polynomial time approximation schemes for euclidean tsp and other geometric problems. pages 2–11. IEEE Computer Society, 1996. 72

A. Atamtürk and M. W. P. Savelsbergh. Integer-Programming Software Systems. *Ann. Oper. Res.*, 140 (1):67–124, 2005. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.104.8697&rep=rep1&type=pdf. 45

J. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numer. Math.*, 4:238–252, 1962. 49

J. Benders. Partitioning procedures for solving mixed-variables programming problems. reprint. *Comput. Manag. Sci.*, 2(1):3–19, 2005. 49

T. Berthold. Primal heuristics for mixed integer programs. Master's thesis, Technischen Universität Berlin, Germany, September 2006. http://opus4.kobv.de/opus4-zib/frontdoor/index/index/docId/1029. 47

T. Berthold. Heuristics of the branch-cut-and-price-framework SCIP. Technical Report 07-30, ZIB, Berlin, 2007. http://opus4.kobv.de/opus4-zib/frontdoor/index/index/docId/1028/. 47

N. Boland and T. Surendonk. A column generation approach to delivery planning over time with inhomogeneous service providers and service interval constraints. *Ann. Oper. Res.*, 108(1-4):143–156, 2001. 56

M. R. Bussieck and S. Vigerske. *MINLP Solver Software.* Wiley, 2011. http://www.math.hu-berlin.de/~stefan/minlpsoft.pdf. 45

W. J. Cook. Concorde TSP Solver. http://www.tsp.gatech.edu/concorde/, 2011. 79

W. J. Cook. *In Pursuit of the Traveling Salesman. Mathematics at the Limits of Computation.* Princeton University Press, 2012. 71, 72, 78, 80

R. J. Dakin. A tree-search algorithm for mixed integer programming problems. *Comput. J.*, 8:250–255, 1965. 39

J. Edmonds. Paths, trees, and flowers. *Canad. J. Math.*, 17:449–467, 1965. http://www.cs.berkeley.edu/~christos/classics/edmonds.ps. 37

U. Feige. A threshold of $\ln n$ for approximating set cover. *J. ACM*, 45(4):634–652, 1998. 70

R. E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bull. Amer. Math. Soc.*, 64:275–278, 1958. 28

R. E. Gomory. An algorithm for the mixed integer problem. Technical Report RM-2597, The Rand Corporation, 1960. 32

M. Grötschel, editor. *Optimization Stories.* German Mathematical Society, 2012. Extra Volume of Documenta Mathematica. 3, 72

K. Helsgaun. An effective implementation of the Lin–Kernighan traveling salesman heuristic. *Eur. J. Oper. Res.*, 126(1):106–130, 2000. 79

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Sci.*, 220(4598): 671–680, 1983. 79

A. A. Korbut and J. J. Finkelstein. *Diskrete Optimierung.* Akademie-Verlag, Berlin, 1971. 3

A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28:497–520, 1960. 39, 41

J. D. C. Little, K. G. Murty, D. W. Sweeney, and C. Karel. An algorithm for the traveling salesman problem. *Oper. Res.*, 11(6):972–989, 1963. 39

F. Margot. Symmetry in integer linear programming. In M. Jünger et al., editor, *50 Years of Integer Programming 1958-2008*, pages 647–686. Springer, Berlin, 2010. `https://student-3k.tepper.cmu.edu/gsiadoc/wp/2008-E37.pdf`. 45

J. S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric tsp, k-mst, and related problems. *SIAM J. Comput.*, 28(4):1298–1309, 1999. 72

Y. Nagata. New EAX crossover for large TSP instances. In T. Runarsson, H.-G. Beyer, E. Burke, J. Merelo-Guervós, L. Whitley, and X. Yao, editors, *Parallel Problem Solving from Nature - PPSN IX*, volume 4193 of *LNCS*, pages 372–381. 2006. 79

G. L. Nemhauser and L. A. Wolsey. *Integer and combinatorial optimization.* Wiley, New York, 1999. 3, 73, 78

M. Padberg and G. Rinaldi. Optimization of a 532-city symmetric traveling salesman problem by branch and cut. *Oper. Res. Lett.*, 6(1):1–7, 1987. 43

C. H. Papadimitriou and S. Vempala. On the approximability of the traveling salesman problem. *Combinatorica*, 26(1):101–120, 2006. 72

J. B. Robinson. On the Hamiltonian game (a traveling-salesman problem). Technical Report RM303, RAND Research Memorandum, The RAND Corporation, 1949. 80

A. Schrijver. *Theory of linear and integer programming. Repr.* Wiley, Chichester, 1998. 3, 14, 63, 64, 76

H. A. Taha. *Integer programming. Theory, applications, and computations.* Academic Press, New York, 1975. 3

L. A. Wolsey. *Integer programming.* Wiley, Chichester, 1998. 3, 54, 62